НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Автоматизованих систем обробки інформації і управління
(повна назва кафедри)

«На правах рукопису»
УДК _____

До захисту допущено:
В.о. завідувача кафедри
_____ Олександр ПАВЛОВ
«___»_____20__ р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-науковою програмою «Інженерія програмного забезпечення комп'ютеризованих систем»

зі спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Математичне та програмне забезпечення виявлення елементів дезінформації в потоках текстових даних»

Виконав (-ла):
студент (-ка) VI курсу, групи ІП-81мн
Ошийко Ярослав Романович _____

Керівник:
Старший викладач
Олійник Юрій Олександрович _____

Рецензент:
Доц. каф. ОТ, к.т.н., доц Артем Волокита _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2020 року

РЕФЕРАТ

Актуальність. З розповсюдженням Інтернету та соціальних медіа зараз доступна кількість новин, статей та іншого тексту онлайн. Цей величезний обсяг інформації постав під загрозу правдивість новин, які поширюються.

Підроблені новини - це будь-яка форма помилкової інформації чи контенту, що поширюється в мережі Інтернет, для впливу на погляд людей на певну подію чи інформацію. Виявлення фальшивих новин у цифровому світі є важливим завданням у подоланні широкого розповсюдження чуток та упереджень. Багато досліджень було проведено для виявлення елементів дезінформації для англійської мови, проте українська та російська мови не має досліджень у цій галузі. Такі компанії, як Facebook, Twitter та Google, стикаються з проблемою вирішення цієї проблеми, щоб забезпечити платформу, де люди можна довіряти вмісту стрічки новин. Вплив фейкових новин було таким глибоко вкорінене в суспільстві, що це навіть вплинуло на вибори в США 2016 року. Також багато неправдивої інформації поширюється протягом війни України в зоні АТО, що призводить до дестабілізації населення, поширення неправильних думок, відображення фейкової картини перебігу подій.

Отже, необхідною задачею є створення інструменту перевірки текстової інформації на наявність елементів дезінформації для інформаційної безпеки та аналізу новин, які поширюються для дестабілізації та обману населення.

Метою дослідження є полегшення виявлення елементів дезінформації за рахунок створення методу та алгоритму для перевірки потоку текстових даних на наявність елементів дезінформації у вигляді лінгвістичних конструкцій та оборотів, які вказують на неправдивість представленої інформації.

Для досягнення поставленої мети необхідно виконати наступні завдання:

- виконати аналіз існуючих алгоритмів та методів комп'ютерної лінгвістики та машинного навчання для класифікації текстових потоків даних та виявлення елементів дезінформації;
- розробити алгоритм первинної обробки тексту для збільшення точності визначення елементів дезінформації;
- розробити метод виявлення елементів дезінформації в текстових потоках даних;
- виконати програмну реалізацію розробленого методу виявлення елементів дезінформації в текстових потоках даних;
- провести аналіз отриманих результатів для оцінки якості;
- провести дослідження ефективності алгоритму.

Предметом дослідження є методи виявлення елементів дезінформації в текстових потоках даних.

Методами дослідження є методи комп’ютерної лінгвістики та машинного навчання для виявлення елементів дезінформації.

Наукова новизна отриманих результатів: науковою новизною є розробка методу виявлення елементів дезінформації в потоках даних з підтримкою обробки текстів української та російської мови.

Публікації: основні положення роботи доповідались і обговорювались на IV всеукраїнській науково-практичній конференції молодих вчених та студентів «Інформаційні системи та технології управління» (ІСТУ-2020), а також подано до друку на XVI міжнародній науковій конференції «Інтелектуальні системи прийняття рішень та проблеми обчислювального інтелекту (ISDMCI’2020)» результати магістерської дисертації докладались на наукових конференціях.

Ключові слова: МАШИННЕ НАВЧАННЯ, КЛАСИФІКАЦІЯ, ДЕЗІНФОРМАЦІЯ, ПОТОКИ ТЕКСТОВИХ ДАНИХ

ABSTRACT

Relevance: With the spread of the Internet and social media, a number of news, articles and other text are now available online. This vast amount of information has jeopardized the veracity of the news being spread.

Fake news is any form of false information or content that is distributed on the Internet to influence people's views on a particular event or information. Detecting fake news in the digital world is an important task in overcoming rumors and prejudices. Many studies have been conducted to identify elements of misinformation for English, but Ukrainian and Russian have no research in this area. Companies such as Facebook, Twitter and Google are facing the challenge of providing this platform to provide people with trust in the content of the news feed. The influence of fake news was so deeply ingrained in society that it even affected the 2016 US election. Also, a lot of false information is spread during the war in Ukraine in the anti-terrorist operation zone, which leads to destabilization of the population, the spread of misconceptions, the reflection of a fake picture of events.

Therefore, a necessary task is to create a tool to check textual information for the presence of elements of misinformation for information security and analysis of news that are disseminated to destabilize and deceive the population.

Purpose: facilitate the detection of elements of misinformation by creating a method and algorithm to check the flow of text data for the presence of elements of misinformation in the form of linguistic constructions and turns that indicate the falsity of the information provided.

To achieve this goal, the following tasks were formulated:

– perform an analysis of existing algorithms and methods of computer linguistics and machine learning to classify text data streams and identify elements of misinformation;

– to develop an algorithm for primary text processing to increase the accuracy of determining the elements of misinformation;

– to develop a method of detecting elements of misinformation in text data streams;

– perform software implementation of the developed method of detecting elements of misinformation in text data streams;

– analyze the results obtained to assess quality;

– to study the effectiveness of the algorithm.

Object of study: methods of detecting elements of misinformation in text data streams.

Research methods: computer linguistics and machine learning methods to identify elements of misinformation.

Scientific novelty: scientific novelty is that the research conducted in the scientific work has the support of the Ukrainian and Russian languages.

Publications: The main provisions of the work were reported and discussed at the IV All-Ukrainian Scientific and Practical Conference of Young Scientists and Students "Information Systems and Management Technologies" (ISTU-2020), as well as at the XVI International Scientific Conference "Intellectual Systems of Decision-making and Problem of Computational Intelligence" (ISDMCI'2020) »The results of the master's dissertation were presented at scientific conferences.

Keywords: MACHINE LEARNING, CLASSIFICATION, DISINFORMATION, TEXT DATA FLOWS.

ЗМІСТ

ВСТУП

З розповсюдженням Інтернету та соціальних медіа сьогодні доступна велика кількість новин, статей та іншого тексту онлайн. Цей величезний обсяг інформації постав під загрозу правдивість новин, які поширюються.

Дезінформація - це будь-яка форма помилкової інформації чи контенту, що поширюється в мережі Інтернет, для впливу на погляд людей на певну подію чи інформацію. Виявлення фальшивих новин у цифровому світі є важливим завданням у подоланні широкого розповсюдження чуток та упереджень. Багато досліджень було проведено для виявлення елементів дезінформації для англійської мови, проте українська та російська мови не має досліджень у цій галузі. Такі компанії, як Facebook, Twitter та Google, стикаються з проблемою вирішення цієї проблеми, щоб забезпечити платформу, де люди можна довіряти вмісту стрічки новин. Вплив фейкових новин було таким глибоко вкорінене в суспільстві, що це навіть вплинуло на вибори в США 2016 року. Також багато неправдивої інформації поширюється протягом війни України в зоні АТО, що призводить до дестабілізації населення, поширення неправильних думок, відображення фейкової картини перебігу подій.

Отже, необхідною задачею є створення інструменту перевірки текстової інформації на наявність елементів дезінформації та аналіз шляхів її розповсюдження для інформаційної безпеки та аналізу новин, які поширюються для дестабілізації та обману населення.

1 ОБЛАСТЬ ЗАСТОСУВАННЯ ТА ЗАГАЛЬНИЙ ОГЛЯД МЕТОДІВ ТА ПІДХОДІВ ДО ВИЯВЛЕННЯ ЕЛЕМЕНТІВ ДЕЗІНФОРМАЦІЇ

1.1 Загальні відомості про дезінформацію та аналіз літератури

Підроблені новини зараз розглядаються як одна з найбільших загроз демократії, журналістиці та свободі вираження поглядів. Вони негативно впливають на погляди суспільства в питаннях політики, воєнних конфліктів відносин між державами. Великий потік дезінформації вплинув на референдум “Brexit” та на вибори США 2016 року. Вплив фейкових новин можна найкраще побачити у останні місяці президентської кампанії на виборах у США, де двадцять найбільш часто обговорюваних фейкових історій отримали 8711000 лайків, репостів та коментарів у Facebook. Що найбільш важливо, більше, ніж 7367000 активностей користувачів були розміщені 19 основними новинними веб-сайтами Америки [1]. Також дезінформація має величезний вплив на економіку, змінюючи бачення на конкретні світові події, тим самим впливаючи на валюту та торги на фондових ринках. Наприклад, фейкові новини, які стверджують, що Барак Обама був постраждалий у результаті вибуху, здійснили падіння вартості акцій на суму більш ніж 130 мільярдів доларів [2]. Ці події та їх наслідки вплинули на важливість дослідження фейкових новин і спричинило появу низки досліджень на цю тему.

Хоча фейкові новини не є новим явищем, питання, як саме вони з'явилося і чому вони привертає все більше уваги громадськості, особливо актуальна в цей час. Основна причина полягає в тому, що підроблені новини можна створювати та публікувати в мережі Інтернет швидше і дешевше порівняно з традиційними новинами в засобах масової інформації, таких як газети та телебачення. Важливу роль у цьому також відіграє поява соціальних мереж та їхній швидкий розвиток. На сьогодні соціальні мережі стали основним плацдармом для публікації та розповсюдження фейкових новин.

Станом на 2019 рік близько двох третин (68%) українців отримують свої новини з соціальних медіа [3]. Крім того, соціальна мережа – це ідеальна платформа для прискорення розповсюдження фальшивих новин, тому що вона вирішує проблему миттєвої передачі інформації на відстані, забезпечує багаті платформи для обміну, пересилання інформації, опитування, перегляду фото та відео інформації, заохочує користувачів для участі в обговоренні онлайн-новин. Ця активність навколо інтернет-новин може призвести до серйозних наслідків, а також до значних політичних та економічних втрат. Ці всі переваги заохочують незаконні організації створювати, публікувати та поширювати фейкові новини. Очевидно, коли уряди, партії та бізнес-магнати стоять за генерацією фальшивих новин, прагнучи влади, впливу та прибутку, є більша мотивація та можливість зробити підроблені новини більш переконливими і такими, що не будуть відрізнятися від правди для публіки. Але, як підроблені новини можуть завоювати довіру громадськості?

Соціальні та психологічні фактори відіграють важливу роль у фальшивих новинах. Ці фактори впливають на довіру громадськості та надалі сприяють цьому поширення фальшивих новин. Дослідження з соціальної психології та комунікацій продемонстрували, що здатність людини виявляти обман лише трохи краща за підкидання монетки: показники точності знаходяться в діапазоні 55%-58%, із середньою точністю 54% при тесті 1000 учасників у понад 100 експериментів [4]. Ситуація є більш критичною для підроблених новин порівняно з іншими видами інформації, тому що для новин, як представників достовірності та об'єктивності, порівняно простіше завоювати довіру громадськості. Крім того, люди схильні довіряти дезінформації після неодноразових повторів (тобто ефект валідності [5]), або якщо вони підтверджують їхні попередні знання (тобто, підтвердження упередженості [6]. Тиск однолітків також може «контролювати» наше сприйняття та поведінку (тобто, ефект пропускної здатності [7]).

Багато поглядів на те, хто створює фейкові новини, як і навіщо вони створюються, як вони поширюються та як дослідити їхнє виявлення та поширення мотивують необхідність глибокого аналізу. Оскільки підроблені новини не визначені чітко, а поточні дослідження фейкових новин обмежені, необхідно дослідити шляхи поширення та правила, за якими створюється дезінформація, та розглянути існуючі методи та дослідження.

1.2 Загальний огляд методів ком'ютерної лінгвістики для виявлення елементів дезінформації

Для вирішення задачі пошуку та розповсюдження фейкової інформації необхідно розуміти принципи, за якими вона поширюється. В цьому допомагають методи комп'ютерної лінгвістики для того, щоб зрозуміти правила та шаблони, за якими можна ідентифікувати елементи дезінформації в потоці текстових даних. Дослідження[8] показали, що для створення фейкової інформації набір тексту використовується за певними правилами, щоб новина здавалася правдивою. Неправдивий контент має певні особливості, такі як скорочення, передача меншої кількості інформації, негативний характер. Також прослідковуються елементи не аналітичної думки, а більш неформального мислення.

Далі приведені дослідження, пов'язані з виявленням дезінформації на основі методів комп'ютерної лінгвістики.

1.2.1 Виявлення спаму як елементу дезінформації

Проблема виявлення фейкових джерел інформації вважається вирішеним в області виявлення спаму[9]. Виявлення спаму використовує статистичні методи машинного навчання для класифікації тексту (твіти або електронні листи), щоб конкретно класифікувати текст як спам або не спам. Ці методи передбачають попередню обробку тексту, визначення ознак (bag of words) та відсікання непотрібних ознак, які призводять до кращої точності на тестовому наборі даних. Коли ці ознаки визначені їх можна класифікувати за

допомогою Naive Bayes, Support Machines, TF-IDF або K-найближчих сусідів. Усі ці класи характерні для навчання з учителем, тобто для них потрібна деяка навчальна вибірка для того, щоб вивчити функцію:

$$f(message, \theta) = \begin{Bmatrix} C_{spam} \text{ якщо визначено як спам} \\ C_{leg} \text{ інакше} \end{Bmatrix}$$

де, m - повідомлення, яке класифікується, і є вектором для параметрів C_{spam} і C_{leg} - відповідно спам та правдиві повідомлення. Завдання виявлення елементів дезінформації подібне до завданням виявлення спаму, тому що в двох випадках стоїть завдання відокремити приклади чесного тексту від прикладів фейкових, неправдивих текстів. Стоїть питання, як ми можемо застосувати подібні прийоми до виявлення елементів дезінформації. На відміну від фільтрації, яка робиться в спамі, було б корисно виділяти статті з елементам дезінформації, щоб читачів можна було попередити, що вони читають швидше за все неправдивий текст. Тому метою є не сама однозначна вказівка на правдивість і неправдивість тексту, а попередження користувача про можливу недостовірність тексту для більш детальної її перевірки. Виявлення елементів дезінформації в тексті, на відміну від виявлення спаму, має багато нюансів, які не так легко виявляються за допомогою аналізу тексту. Наприклад, новина може стати фейковою замінивши назву однієї людини на іншу, тому простий аналіз тексту не допоможе у визначення правдивості чи неправдивості тексту. Тому для кращої точності визначення тексту модель треба навчати на основі величезного масиву інших новин та статей по конкретній темі. В цьому аспекті пошук шляху розповсюдження фейкових новин допомагає співставити попередню інформацію і перевірити конкретних особистостей чи фактів.

1.2.2 Синтаксичний аналіз

Оскільки аналізу слів недостатньо для передбачення фейкових новин, можна використовувати інші мовні підходи, такі як аналіз синтаксису та

граматики мови. Використовуються Probability Context-Free Grammars (PCFG) [10] для перетворення речень у дерева розбору, що описують структуру речень та для перетворення речення до рекурсивної синтаксичної структури. Синтаксичний аналіз широко використовується для аналізу настрою тексту (семантичний аналіз).

Під час синтаксичного аналізу текст оформлюється у структуру даних, зазвичай в дерево, яке відповідає синтаксичній структурі вхідної послідовності, і добре підходить для подальшої обробки. Зазвичай синтаксичні аналізатори працюють в два етапи: на першому ідентифікуються осмислені токени (виконується лексичний аналіз), на другому створюється дерево розбору.

Синтаксичний аналізатор — це програмний компонент, який приймає вхідні дані (в нашому випадку текст) і створює структуру даних, таку як дерево розбору, абстрактне дерево синтаксису або іншу ієрархічну структуру, що забезпечує структурне представлення вводу, перевіряє правильність синтаксису в процесі. Для аналізу попередньо виконується обробка тексту, тобто видалення стоп слів, нормалізація, стемінг тощо. Аналізатору часто передує окремий лексичний аналізатор, який створює токени з послідовності введених символів. Аналізатори можуть бути запрограмовані вручну або автоматично[11].

1.2.3 Семантичний аналіз

У більшості випадків правдивість певної новини чи тексту можна передбачити, вивчивши коментарі та подібні статті. Якщо більшість подібних статей не відповідають новинам, найімовірніше, що новини можуть бути упередженими або фальшивими. Аналогічно коментарі до статті можуть бути використані для оцінки того, чи є факти у статті достовірні.

Підхід до семантичного аналізу - це як розширення моделі n-граму разом з синтаксичною модель та з функціями сумісності профілю.

Основні недоліки такого підходу включають складність автоматичного пошуку подібних статей, перевірки сумісність профілю та облік різних слів, які мають на увазі те ж саме.

1.2.4 Знаходження дезінформації за допомогою мережевих методів

Інтернет містить величезну кількість метаданих, які можна використовувати для прогнозування надійності джерела. Twitter, Facebook та Google має великий мережевий набір даних, пов’язаний з кожним користувачем[12]. Ці дані і можуть використовуватися в мережевих методах. Розділяються два підходи:

- аналіз метаданих;
- перевірка пов’язаних даних і фактів.

Розглянемо ці два методи.

Метадані. Мережевий підхід вивчає метадані, наприклад URL-адреси, автори, вподобання у соціальних мережах тощо і прогнозується висновок чи джерело надійне. Модель можна навчити так, щоб вона враховувала метадані сайту в поєднанні з особливостями, створеними в результаті аналізу настроїв. Використання цих двох різних критеріїв дає зможу отримати кращу точність визначення дезінформації в потоках текстових даних на тестовому датасеті.

Перевірка пов'язаних даних та фактів. Правдивість новин можна також визначити, перевіривши факти, згадані в новинах. Генерування дерева відношення для перевірки фактів є одним із підходів для цього. Перевірка фактів - це лише спосіб використання знань мережі для перевірки фактів. Певні фактичні твердження, що містять дані або відносини можна перевірити, використовуючи надійні джерела. Хоча перевірка фактів має ряд переваг у правильній класифікації фальшивих новин, процес перевірки важко автоматизувати, що призводить до низької швидкодії цього методу.

1.2.5 Дослідження фейків за допомогою онлайн-інструментів

Дослідники [13] американської мережі Public Data Lab запропонували скористатися онлайн-інструментами CrowdTangle, Google News Search та Gephi для відстеження траєкторії поширення фейкових новин в соціальній мережі Facebook. На основі свого дослідження вони зробили висновки, як дослідити траєкторію поширення фейкової новини.

1) Необхідно визначити, які типи користувачів залучають фейкові новини у Facebook.

На цьому етапі необхідно виконати такі кроки:

– ідентифікувати теми, які експлуатуються в потоці фейків;

– визначити, які відомі публічні сторінки та групи у Facebook поширювали ці історії;

– розглянути, чи є в цієї аудиторії улюблені теми, до яких вона звертається частіше;

– підсумувати зібрану інформацію та скласти збірний образ аудиторії, яка охоче приймає та поширює фейки.

Дослідники використали для прикладу 22 лінки на фейкові статті. Вони проаналізували їх за допомогою розширення для браузера CrowdTangle [13]— цей інструмент дозволяє відстежувати, як контент поширюється в мережі та наскільки ефективно він залучав аудиторію на Facebook, Twitter, YouTube, Instagram та Vine. Потім вони виокремили на таймлайні той проміжок, коли фейки публікувалися з найменшим інтервалом часу. Після цього вони почали шукати ключові релевантні події — проаналізувавши новини за допомогою Google News Search, які були актуальними в той же час. Портал Gephi дозволяє знаходити спільні теми і тренди та потім візуалізовувати їх (рисунок 1.1).

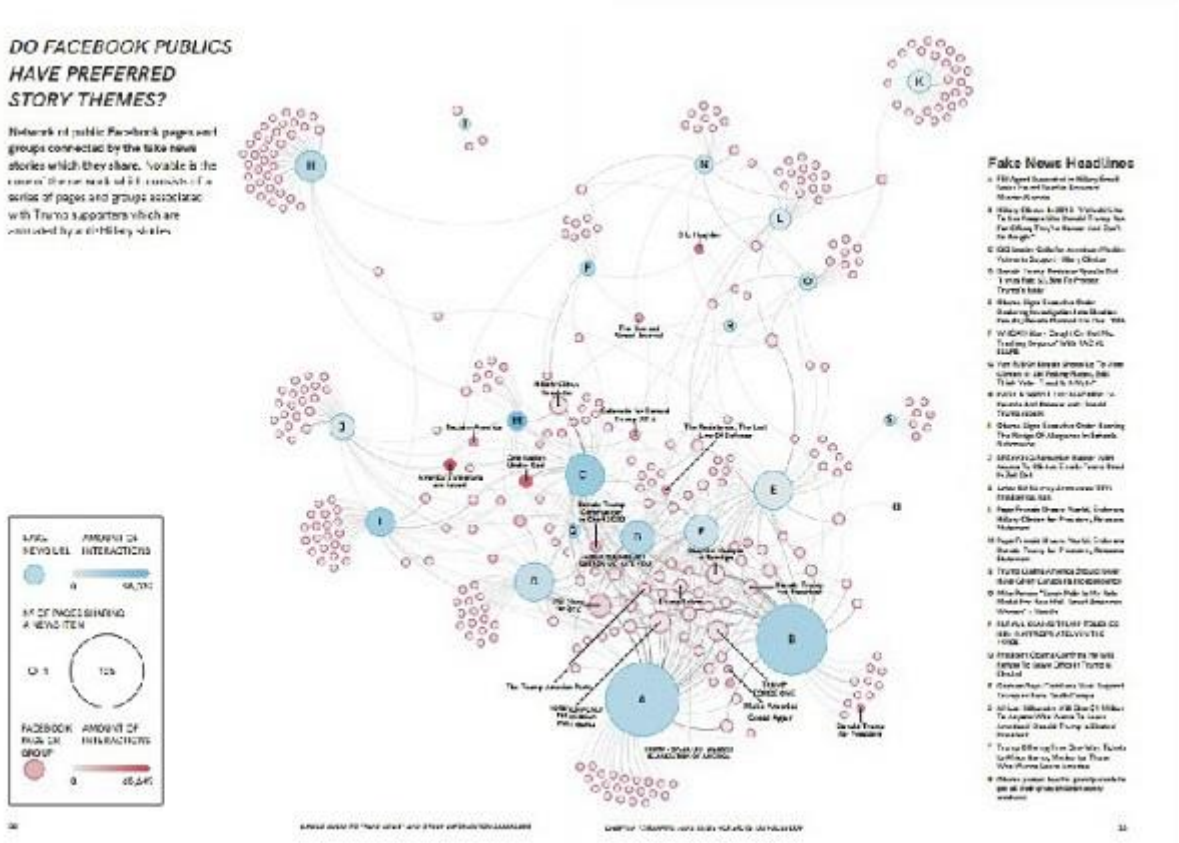

Рисунок 1.1 – Візуалізація спільних тем на порталі Gephi

2) Відстеження траєкторії поширення новин на Facebook

В цьому питанні дослідники дослідили дві проблеми:

– відстеження окремої сторінки, яка поширила лінк на новину;
– відстеження всіх сторінок, які поширили певний лінк.

В цьому допомагає розширення для браузеру CrowdTangle. Для обробки результатів рекомендується створити таймлайн за допомогою RAWGraphs.

Також в пошуку траєкторії допомагає Google Web Search[14]. Зібравши усі лінки на новини в Google Web Search, їх можна перевірити за допомогою CrowdTangle й побачити, хто поширював ці лінки в Facebook.

3) Необхідно перевірити, чи фактчекери охоплюють аудиторію поширювачів фейкових новин у Facebook

Тут рекомендується відокреми лінки на сам фейк та на його спростування. Потім потрібно проаналізувати поширення цих новин через

CrowdTangle. Метою є перевірки, чи сторінка, яка поширила фейк, потім поширила і його спростування.

Аналіз показав, що користувачі, які читають та поширюють фейки, та користувачі, які читають та спростовують фейки – це різна аудиторія.

Цим методом можна перевірити ефективність порталів перевірки фактів.

4) Пошук найбільш популярних джерел фейків

Для цього через Google отримуємо перелік посилань, які поширювали цю новину, а також посилання, які спростували її. Важливо фіксувати, які сторінки цитували одна одну. Для всіх публікацій потрібно окремо фіксувати дату. Усі зібрані дані потрібно внести в таблицю (формат .csv) й потім за допомогою Table2Net трансформувати її у файл із розширенням .gexf. Цей останній файл потрібно імпортувати в Gephi — сервіс створить візуалізацію на основі даних із вашої таблиці.

Зібрані дані можна так само перетворити на таймлайн у Graph Recipes, аби зрозуміти часові рамки того, як поширювався фейк і його спростування. Цей рецепт буде корисний для того, аби під новим кутом поглянути на зібрані дані й проаналізувати їх з різних боків. Він також показує зв’язки між різними сайтами й те, як фейк поширюється, з яких видань до яких.

1.2.6 Перевірка фактів як елемент виявлення дезінформації

Вивчаючи підроблені новини з точки зору знань, потрібно прагнути проаналізувати та виявити підроблені новини, використовуючи процес, відомий як перевірка фактів. Перевірка фактів, яка спочатку використовувалася в журналістиці, має на меті оцінити справжність новин за допомогою порівняння знань, отриманих із контенту новин, що підлягає підтвердженню з відомими фактами. У цьому розділі описано традиційну перевірку фактів (також відому як ручна перевірка фактів) і автоматичні методи для аналізу та виявлення фальшивих новин (тобто автоматичної перевірки фактів).

1.2.7 Мануальна перевірка фактів

Загалом, ручну перевірку фактів можна поділити на експертну та краусорсингову перевірку. Експертна перевірка фактів[15][16][17] покладається на експертів певного середовища перевірки фактів для перевірки необхідного змісту контенту. Експертна перевірка фактів найчастіше проводиться для перевірки невеликої групи даних, найчастіше конкретного факту. Цей спосіб легкий в використанні і має високу точність у перевірці, проте є дорогим і погано масштабованим при збільшенні кількості фактів, які необхідно перевірити. У таблиці 1.1 наведено порівняння веб-сайтів, які надають послуги такої перевірки.

Таблиця 1.1 – Порівняння експертних веб-сайтів перевірки фактів англійською мовою

Назва	Охоплені теми	Контент, який аналізується	Оцінки, які отримуються
PolitiFact	Американська політика	Заяви	Правда; Здебільшого вірно; Напівправда; Переважно хибні; Помилковий;
The Washington Post Fact Checker	Американська політика	Заяви та претензії	Один піноккіо; Два піноккіо; Три піноккіо; Чотири піноккіо; Галочка Geppetto; Пінокіо перевернутий; Вирок не очікується

Продовження таблиці 1.1

FactCheck	Американська політика	Реклама, дебати, промови, інтерв'ю, новини	Правда; Ніяких доказів; помилковий
Snopes	Політика та інші соціальні теми	Новини та відео	Правда; Здебільшого вірно; Суміш; Переважно хибні; Помилковий; Недоведений; Застарілий; Помилково; Правильна атрибуція; Нерозподілений; Шахрайство; Легенда
TruthOrFiction	Політика, релігія, природа, авіація, їжа, медицина тощо	Чутки по електронній пошті	Істина; Художня література; Помилковий
FullFact	Економіка, здоров'я, освіта, закони, злочини	Статті	Неоднозначність (відсутні чіткі мітки)
HoaxSlayer	Неоднозначності	Статті і повідомлення	Підман, шахрайство, гумор, спами тощо

Краусорсингова перевірка фактів акумулює в собі перевірки фактів на багатьох середовищах. Порівняно з експертною перевіркою, краусорсингова перевірка фактів менш налаштовувана, є менш достовірною та точною через політичну упередженість перевіряючих фактів і їхні суперечливі примітки та мають кращу (хоча й недостатню) масштабованість. В отриманій інформації необхідно фільтрувати невідомі джерела та вирішувати конфліктні результати. В результаті обидві вимоги стають більш критичними, оскільки кількість перевіряючих фактів зростає. Тим не менш, ці платформи часто надають більш детальний зворотний зв’язок для фактів, які перевіряються (наприклад, їх настрої чи позиції). Ці результати можна додатково проаналізувати в дослідженнях підроблених новин.

На відміну від перевірки фактів на основі експертів, краусорсингова перевірка фактів знаходиться ще в ранньому розвитку. Приклад - Fiskkit[18], де користувачі можуть завантажувати статті, надавати рейтинги для пропозицій всередині статей і вибирайте теги, які найкраще описують його. Наведені джерела статей допомагають розрізняти типи змісту (наприклад, новини чи не новин) та визначають її достовірність. Теги, які класифіковані на кілька параметрів, дозволяють вивчити закономірності в підроблених та неправдивих новинних статтях (див. рисунок 1.2).

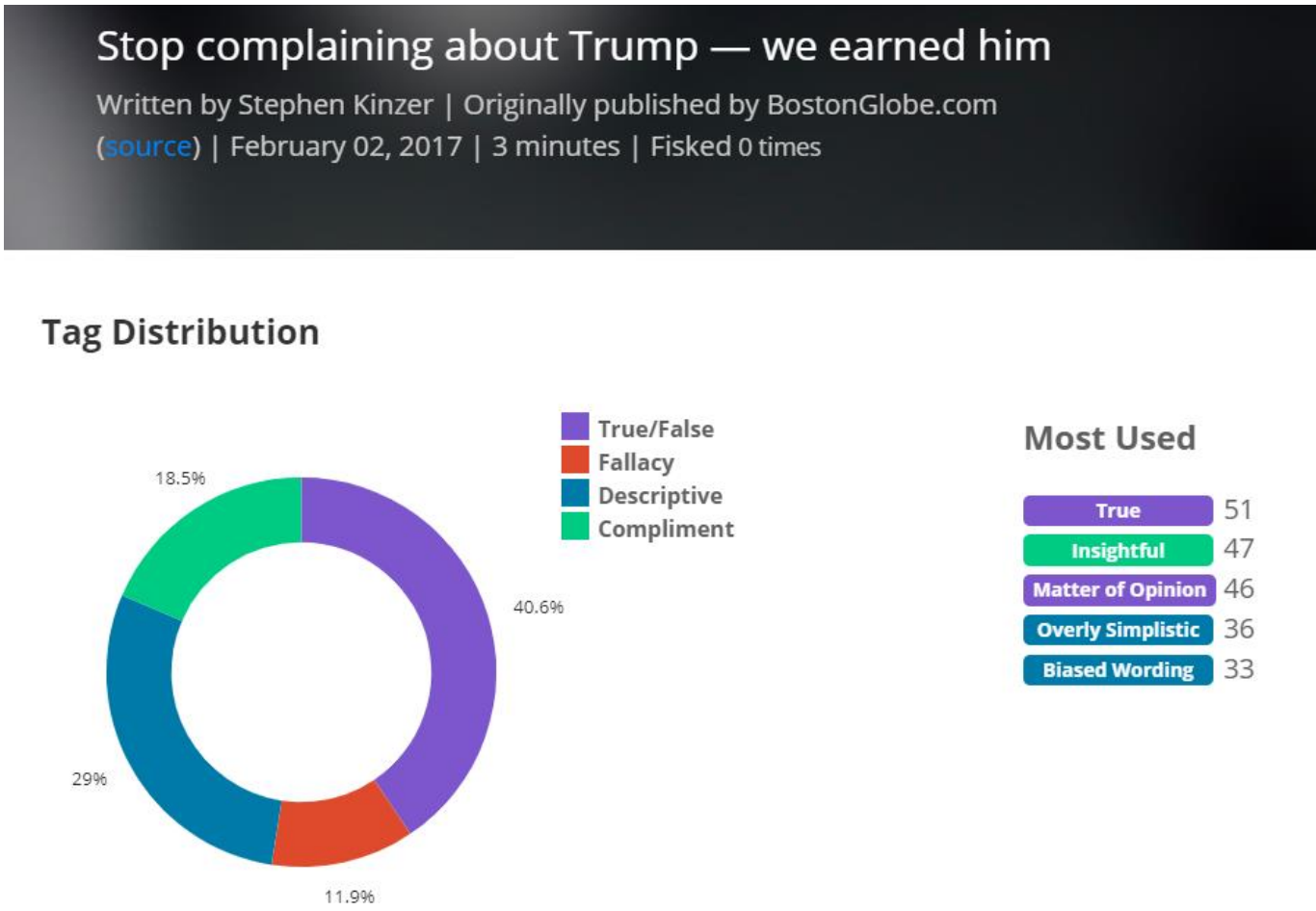

Рисунок 1.2 - Розподілення тегів Fiskkit

Незважаючи на те, що таких веб-сайти не так багато, вважається, що більше платформ або інструментів, виникне найближчим часо, оскільки основні веб-сайти та соціальні мережі усвідомили важливість визначення фальшивих новин (такі як Google, Facebook, Twitter).

1.2.8 Автоматична перевірка фактів

Ручна перевірка фактів не встигає за кількістю нової інформації, особливо в соціальних медіа. Для цього розроблені автоматичні методи перевірки фактів на основі пошуку інформації (IR) та обробки природної мови (NLP).

Загальний процес автоматичної перевірки фактів можна розділити на два етапи: вилучення фактів (також відоме як побудова бази знань) та перевірка фактів (також відоме як порівняння знань). Отримання інформації

видобувається часто з мережі Інтернет, який надає масивну неструктуровану інформацію у вигляді онлайн-документів. Витягнуті знання використовуються для побудови бази знань (КБ) або графіка знань, кожен з яких містить набір фактів (тобто справжні знання) після належного очищення даних. Під час перевірки фактів достовірність змісту новин, що підлягають підтвердженню, визначається шляхом порівняння знань, витягнутих із змісту новин, з фактами, що зберігаються в побудованій базі знань або графіку знань.

Для збору фактів дані часто витягуються з відкритої мережі Інтернет як "необроблені факти", тобто знання, які є зайвими, застарілими, конфліктними, ненадійними або неповними. Ці необґрунтовані факти додатково обробляються та очищаються за допомогою попередньої обробки тексту, щоб створити базу знань або графік знань. Видобуток знань має на меті зібрати необґрунтовані факти з відкритої Мережі. Загалом, існує чотири типи веб-контенту:

– текст;
– табличні дані;
– структуровані сторінки;
– людські анотації.

Ці дані містять реляційну інформацію та можуть використовуватися для отримання фактів різними парсерами[19].

Щоб оцінити справжність новинних статей, нам потрібно додатково порівняти отримані факти, що підлягають підтвердженню з фактами, що зберігаються у створеній або існуючій базі фактів, тобто справжньої інформації. Для цього використовуються лінгвістичні методи, описані в розділах 1.2.1 – 1.2.4.

1.3 Огляд правил, за якими створюється та розповсюджується дезінформація

Термін «фейк» сьогодні використовується в багатьох мовах. Синонімами до цього терміну є підробка, вигадка, шахрайство. Концент фейкової інформації включає декілька різних медіа феноменів, таких як:

– фейкові тексти, фото, відео та аудіо файли;

– фейкові сторінки та блоги в соціальних мережах, створені на основі реальних людей, місць, подій, історичних та літературних персонажів;

– фейкові сторінки популярних веб-сайтів (найчастіше використовуються для викрадення інформації користувача про реєстраційні дані з реального веб-сайту);

– стаття чи доповідь під виглядом реального автора, яку фейк-боти розповсюджують за допомогою лайків та коментарів;

– повністю фейкова або частково спотворена інформацію про події та факти.

Аналіз значної кількості фейкових новин дає змогу розрізнити наступні типи дезінформації[20]:

– з точки зору співвідношення правдивої та хибної інформації;

– новина може бути повноцінною брехнею.

Зазвичай, це новини пов’язані з повною видумкою факту. Також популярним різновидом цього типу є фейкові новини про смерть знаменитості. Часто новину присвоюють людині, яка це не говорила. Прикладом такої фейкової новини є інформація про те, що коронавірусу в Африці та Індії не існує[10]. Стверджувалося, що коронавірус нібито не торкнувся африканського континенту через те, що в Африці «щільність населення є нижчою від китайської і немає ніяких засобів швидкого переміщення величезних мас людей», а мешканці Індії «мають імунітет». Ця новина почала поширювати наприкінці березня, хоча перші випадки в Африці датувалися ще в лютому в Єгипті, а в Індії ще в кінці січня. На сьогодні в Африці підтверджено близько 11 тисяч випадків, в Індії 5 тисяч. Приклад новини наведено на рисунку 1.2.

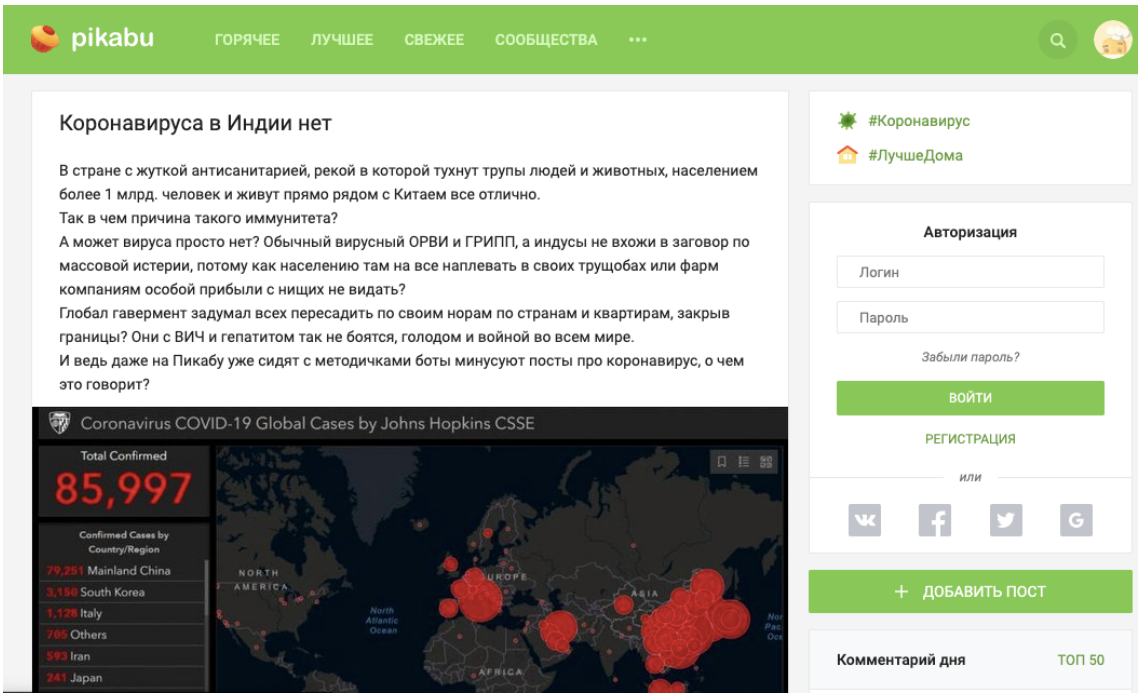

Рисунок 2.3 – Приклад повноційної фейкової новини

– Новина містить брехню, тоді як вся інша інформація у цій новині є достовірною, але представлена вибірково

Прикладом такої новини є поширення інформації, що у США знайдено та заарештовано творця коронавірусу[20] (рисунок 1.3).

Рисунок 1.4 – Приклад фейкової новини з частковою правдою

Незважаючи на гучний заголовок, у самій статті зазначено, що Чарльз Лібер навряд чи був причетним до розробки коронавірусу – натомість автор стверджує, що науковець, найвірогідніше, «зливав» Китаю американські військові розробки.

Ще в лютому портал перевірки фактів Snopes опублікував дослідження[21], де описав, що насправді ж гарвардського професора заарештували ще наприкінці січня, і офіційно Чарльза Лібера звинувачують в тому, що він свідомо надавав Міністерству оборони США і Національному інституту охорони здоров'я США неправдиву інформацію щодо своєї співпраці з Уханським університетом технологій і участі в китайській дослідницькій програмі "Thousand Talents Plan" впродовж 2012 – 2017 років. В Уханському університеті він організував також спільну лабораторію Гарвард-Ухань, на створення якої отримав грант від університету і китайського уряду на суму 1,5 млн доларів, не узгодивши цього з керівництвом Гарварду, чим порушив корпоративну політику. Він також отримував регулярні виплати від програми "Thousand Talents Plan" і Уханського університету технологій, які приховував від уряду США.

– Підроблені новини на основі реальної події, деталі якої неправильно представлені.

Цей тип підроблених новин може бути аудіо, відео файлом чи фотографією, відредаговану таким чином, як створювач фейку захотів його відредагувати. Іншим прикладом є цитати, взяті з їх початкових контекстів і, наприклад, представлені аудиторії у певному порядку.

– З точки зору достовірності часу і місця події.

– Подія, яка трапилася в минулому, публікується як сучасна.

Це може бути певна угода між країнами чи певна історична подія, яка представлення так, наче вона відбулося сьогодні.

– Частина новини про події в конкретному місці представлена як подія, що трапилася в іншому місці.

Прикладом такої новини є відео, яке поширювалося в Ірані[22]. Нібито місцева поліція під час протестів грабувала магазини для власного прибутку. Проаналізувавши відео та надписи на ньому стає очевидним, що це було відео, яке було знято пару років тому в Мехіко.

– З точки осіб, які згадуються в новинах.

– Новина стосується заяви офіційної особи, опублікованої на підробленій веб-сторінці особистих.

– Новина, де периферійний персонаж представлений як головний учасник події.

– Новини, засновані на неперевірених свідченнях тих, хто претендує бути свідками певних подій.

– З точки зору цілей, які переслідують творці фейкових новин

– Новина, створена для розваги глядачів.

Сучасне медіа-середовище включає окрему нішу інформаційних ресурсів, створених для публікації фальшивих новин. Один з найбільш відомих інформаційних ресурсів США, який створює сатиричні фейкові новини, це медіа-проект під назвою The Onion[23]. Версія для паперу була закрита в 2013 році, і нині Theion також транслює відео новини на YouTube. Інтернет-телеканал називається ONN (Onion News Network) - за аналогією до CNN. Крім підроблених новин, канал транслює фальшиві ток-шоу, фальшиві розслідування та останні огляди фільмів. В Україні прикладом такого інформаційного ресурсу є

– Фальшиві новини, створені та розповсюджені для отримання політичних переваг, дискредитування політичних опонентів (що включає дискредитацію їх під час виборчих кампаній), провокування заворушень чи перевороту тощо.

Наприклад, під час президентської виборчої кампанії в США вони опублікували підроблені новини про Хілларі Клінтон та Білла Клінтон, які ведуть злочинний бізнес по залученню до сексуального насильства над дітьми через піцу-ресторан «Комета Пінг Понг» через те, що власники ресторану не намагалися запобігти злочину. Як результат, ресторан почав отримувати тисячі погроз, нові скандальні історії розповсюджувались кожного дня, а на подвір'ї ресторану вони виявили жінок, які шукають таємні тунелі, які будують для передачі дітей для занять проституцією та магічними ритуалами. У грудні 2016 року в одному з ресторанів «Комета Пінг Понг» у Вашингтоні 28-річний чоловік почав стріляти. Свідки казали, що він стріляв, стверджуючи, що готовий розслідувати так званий Піцагейт. Потрібно відзначити, що Дональд Трамп, кілька разів звільнив одного з членів своєї перехідної команди Майкла Фліна-молодшого за те, що він публікував схвальні твіти про Піцагейт.

– Підроблені новини, створені з метою дискримінації людей через їхню стать, расу, національність, мову, соціальний статус, місце проживання, ставлення до релігії, поглядів, належність до неурядової організації тощо.

Показовим прикладом цього можуть бути медіа-війни, які ведуться поряд із справжні війни в проблемних районах світу.

– Фальшиві новини створені та розповсюджені для маніпулювання ринком для отримання економічних переваг.

У листопаді 2016 року акції французької будівельної компанії Vinci SA знизилися на двадцять відсотків, компанія втратила близько семи мільярдів євро акціонерного капіталу. Це сталося після того, як кілька фінансових ЗМІ опублікували фальшиві новини про звільнення директора фінансової компанії. У підробленому прес-релізі зазначається, що топ-менеджера звільнили через помилки, які він допустив у своїх доповідях, оскільки ці помилки могли спричинити завищення результатів компанії на 3,5 мільярда євро. У прес-релізі містилася посилання на веб-сайт, що імітує реальний веб-сайт

будівельної компанії. Він також містив підроблений номер телефону, який нібито належав прес-секретарю Вінчі Паулю Алексісу Буке. Чоловік, відповідаючи на дзвінки журналістів Dow Jones Newswires, представився як Пол-Алексіс і підтвердив цю новину. Потім на тому ж веб-сайті недоброзичливці опублікували часткове спростування цієї новини, що призвело до ще більшої плутанини серед гравців ринку.

– Фальшиві новини створені та розповсюджені для збільшення Інтернет-трафіку.

– Підроблені новини створені для викрадення майна.

– Підроблені новини, створені для знищення інформація, що зберігається чужому комп’ютері

Наприклад, фальшиві новини про смерть Бреда Пітта[25], нібито опубліковані американським телеканалом Fox News, виявилися комп'ютерним вірусом, який запускався користувачем при натисканні на цю новину.

– Підроблені новини, створені для привернення уваги аудиторії до конкретної людини, компанії, проекту чи руху.

– З точки зору ступеня достовірності новин

– Очевидно фальшиві новини.

– Підроблені новини, які не видаються явно фальшивими, і читач відчуває необхідність перевірити опубліковану інформацію.

– Підроблені новини, написані настільки переконливо, що у читача немає сумнівів у тому, що вони є автентичною новиною.

1.4 Висновки до розділу

У розділі було описано дезінформацію як явище, описано її класифікацію, шляхи та причини її створення, проаналізовано її розповсюдження. Було показано вплив її на світову політику та економіку.

Основною метою поширення фейків є створення конфліктних ситуацій, дискримінація певної особистості чи події, отримання економічної вигоди.

Розглянуто лінгвістичні та мануальні методи перевірки інформації на достовірність фактів. Портали перевірки інформації, такі як The Washington Post Fact Checker, Snopes, StopFake допомагають перевірити факти і встановити є новина правдивою.

Описано шляхи поширення фейкових новин, мету їх створення.

Задачу наукової роботи можна сформулювати як дослідження шляху розповсюдження фейкової інформації для швидкого знаходження першоджерела. Пропонується розробити алгоритм знаходження першоджерела фейкової новини та отримання шляху перетворення новини в часовому інтервалі.

2 МАТЕМАТИЧНА МОДЕЛЬ ГЕНЕРУВАННЯ ШЛЯХУ РОЗПОВСЮДЖЕННЯ ФЕЙКОВИХ НОВИН

Після розгляду загальних принципів розповсюдження дезінформації, в науковій роботі поставлена наступна задача: пошук першоджерела фейкової новини та шляху її розповсюдження. Для її вирішення необхідно вирішити наступні задачі:

- Згенерувати набір з новинами.
- Виконати класифікацію фейкових новин.
- Виконати первинну обробку текстів новин.
- Застосувати алгоритм визначення подібності двох текстів.
- Побудувати дерево розповсюдження фейкової новини.

Для виконання задачі було згенеровано набір даних на основі українського сайту з виявлення фейків StopFake.org. Було отримано вміст фейкових новин з різних ресурсів, тематику новин та дату її створення. Після успішного збереження новин була проведена первинна обробка текстів нових.

2.1 Первинна обробка тексту

Базуючись на дослідженні [26] первинна обробка тексту необхідна збільшення точності обчислення схожості новин. Обробка тексту поділяється на такі етапи:

- Приведення слів до нижнього регістру;
- Стемінг;
- Лематизація;
- Видалення стоп-слів;
- Нормалізація;
- Видалення шуму;

Розглянемо детально кожен етап попередньої обробки тексту.

Приведення всіх слів до нижнього регістру, хоча і зазвичай не помічається, є однією з найпростіших та найефективніших форм попередньої обробки тексту. Цей метод застосовується до більшості проблем із обробкою тексту та проблемами NLP і може допомогти у випадках, коли ваш набір даних не дуже великий і суттєво допомагає забезпечити узгодженість очікуваного результату.

Стеммінг[47] - це процес приведення слів (наприклад, неприємностей) до їх кореневої форми (наприклад, неприємності). "Корінь" у цьому випадку може бути не справжнім кореневим словом, а лише канонічною формою очаткового слова.

Іноді алгоритми лематизації теж мають стохастичні властивості, коли частину мови вони визначають без урахування контексту, в якому це слово було вжито в реченні. У таких випадках перевага віддається найвірогіднішій частині мови для цього слова, і як результат - ймовірність помилок стемінгу зростає.

Лематизація на поверхні дуже схожа на стемінг, мета якого це приведення слово до кореневої форми. Різниця лише в тому, що лематизація намагається зробити це належним чином. Це не просто відсікає закінчення, а фактиче перетворення слова у фактичний корінь.

Видалення стоп-слів. На цьому етапі видаляються всі слова, які не несуть змістовного характеру. До цих слів належать:

– Спілки та союзні слова (або, але, щоб, потім, потім, як тільки).

– Займенники (він, ми, його, ви, вам, вас, її, що, який, їх, все, вони, я, весь, мені, мене, таким).

– Причини того (для, на, по, з, з, від, до, без, над, під, за, при, після, під).

– Частинки (ні, ж, то, б, все, все, навіть, так, ні).

– Вигуки (ой, ого, ех, браво, здрастуйте, спасибі, вибачте).

– Цифри і числівники (1, 2, 3 один, два, три перший, другий, третій тощо).

– Розділові знаки і спеціальні символи (., - _ = + /!;:%? *).

– Вступні слова (скажімо, може, припустимо, чесно кажучи, наприклад, насправді, однак, взагалі, в загальному, ймовірно).

– Окремо стоять літери (а, б, в тощо).

– Невизначені приводи, прислівники і деякі звичайні прислівники (щось, якийсь, десь, якось, далі, ближче, раніше, пізніше, коли-то).

– Слова-підсилювачі (дуже, мінімально, максимально, абсолютно, величезний, гранично, сильно, слабо, самий).

– Ряд деяких іменників, дієслів, прислівників (наприклад, сайт, давати, завжди, однак і інші).

Процес нормалізації дозволяє прибрати з тексту граматичну інформацію (відмінки, числа, дієслівні види і часи, дієприкметники, рід і так далі).

2.2 Алгоритм класифікації дезінформації

Для класифікації дезінформації в потоках текстових даних використовується синтиментальний аналіз попередньо обробленого тексту та подальша перевірка фактів за допомогою сервісів описаних в розділі 1.2.7.

Першим етапом в алгоритму виявлення дезінформації буде класифікації текстів за синтиментом. Метою цього алгоритму є визначення, до якого класу належить текст: негативний чи позитивний. Після отримання класу тексту, можна відкинути частину текстів, які не мають негативних елементів, звідки можна зробити висновок про відсутність дезінформації.

Після первинної обробки тексту необхідно побудувати корпус ознак, який базується на моделі Мішок слів (Bag of words) та метриці TF-IDF.

Bag of words[27] - це представлення слів по відношенню до їх частот в тексті. Мішок слів схожий на об'єкт JSON, що містить список слів як ключі та частота слова як значення. Використовуючи одиничні слова і n-грам, що

переважають у дезінформованому тексті, цей підхід може перевірити домінування таких слів у тексті і зробити висновок, чи є матеріал фейковий.

Простий bag of words встановлює велику частоту для поширених слів, таких як займенники (я, він, вона, вони), сполучників тощо. За допомогою TF-IDF[28] долається цю неефективність шляхом змешення значення частоти слів, які сильно вживаються в документ. Це забезпечує краще подання частоти слів у документах, які були трансформовані.

Обмеженням підходу bag of words полягає в тому, що він заснований на використанні слів у документі без урахування будь-якої інформації про контекст.

TF-IDF[28] (TF — term frequency, IDF — inverse document frequency) – це статистичний показник, який оцінює важливість слів в контексті тексту. Цей показник використовується в задачах аналізу тексту та інформаційного пошуку.

TF (term frequency — частота слова) — відношення числа входжень обраного слова до загальної кількості слів документа. Таким чином, оцінюється важливість слова t_i в межах обраного документа.

$$TF = \frac{n_i}{\sum_k n_k}$$

де n_i є число входжень слова в документ, а в знаменнику — загальна кількість слів в документі.

IDF (inverse document frequency — обернена частота документа) — інверсія частоти, з якою слово зустрічається в документах колекції. Використання IDF зменшує вагу широковживаних слів.

$$IDF = \log \frac{|D|}{|(d_i \ni t_i)|}$$

де $|D|$ — кількість документів колекції, $|(d_i \ni t_i)|$ — кількість документів, в яких зустрічається слово t_i (коли $n_i \neq 0$).

Вибір основи логарифму у формулі не має значення, адже зміна основи призведе до зміни ваги кожного слова на постійний множник, тобто вагове співвідношення залишиться незмінним. Іншими словами, показник TF-IDF це добуток двох множників: TF та IDF.

$$TF - IDF = TF * IF$$

Більшу вагу TF-IDF отримають слова з високою частотою появи в межах документа та низькою частотою вживання в інших документах колекції.

У визначенні елементів дезінформації у потоці текстових даних TF-IDF дає можливість визначити важливість кожного слова в тексті. Проте важливо визначити і відповісти чи текст містить дезінформацію чи ні. Тому алгоритм використовує статистичні показники отримані в результаті обчислення TF-IDF і прогнозує чи конкретне слово відповідає реальній чи фейковій новині.

Після отримання корпусу ознак відбувається класифікація тексту за синтиментом. Найпоширенішим алгоритмом для цього є градієнтний бустинг. Це технологія машинного навчання для задач регресії та класифікації, яка дозволяє створити медель в форматі ансаблю слабких моделей. В цій технології використовується дерево прийняття рішень.

Після визначення класів текстів виділяються тексти з негативною тональність і запускається процес перевірки фактів за допомогою фактчекінгових сервісів.

Загальний алгоритм класифікації зображений на рисунку 2.1.

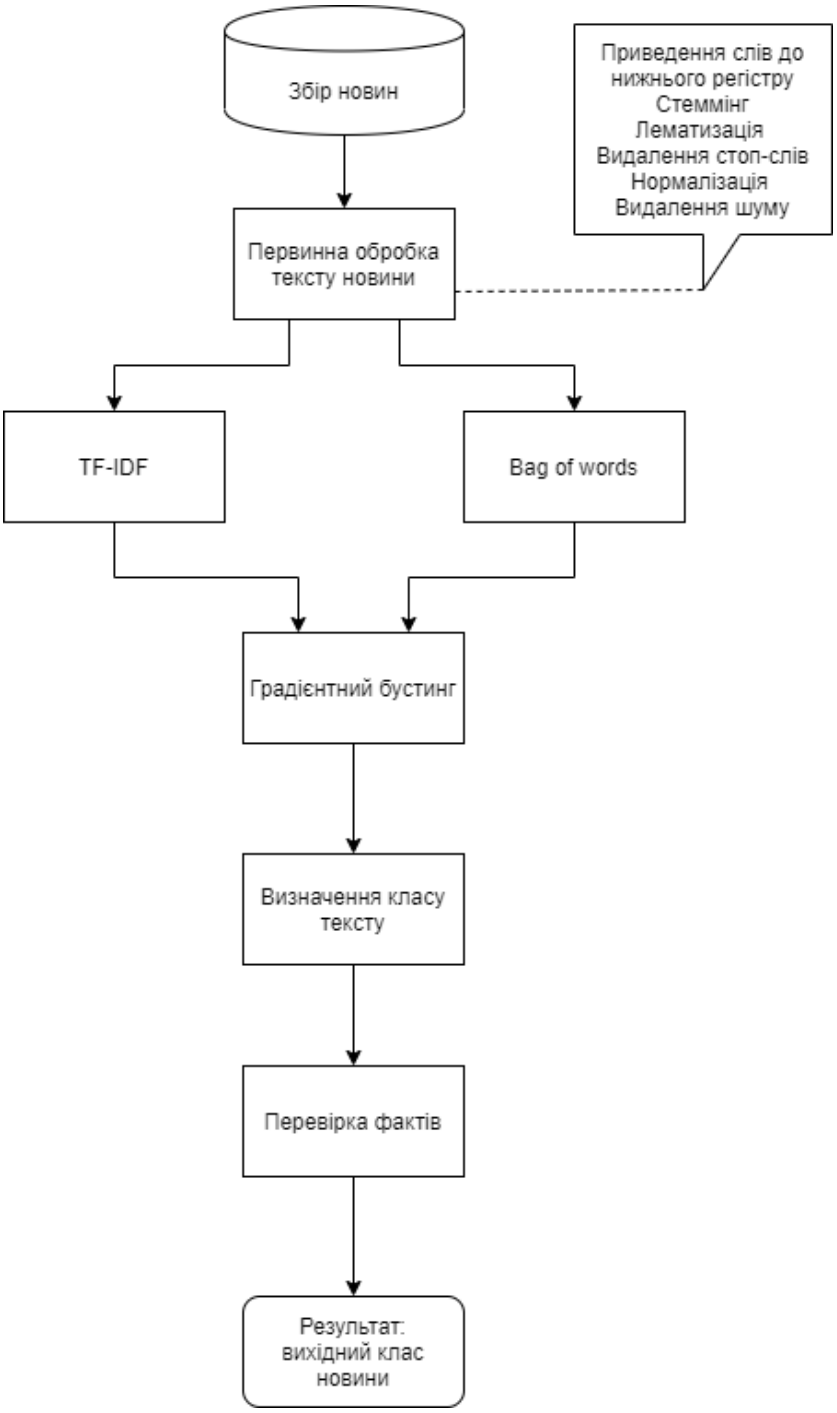

Рисунок 2.1 – Процес класифікації тексту новин на наявність дезінформації

2.3 Алгоритм визначення схожості фейкових новин

Для визначення схожості фейкових новин розглянуто дві метрики: коефіцієнт Жаккара та косинус подібності. Для обчислення цих показників необхідно виділити додаткові ознаки тексту та перевести їх в векторний вигляд. Для цього використовується модель Мішок Слів та метрика TF-IDF.

Коефіцієнт Жаккара[30] – це показник подібності текстових документів. Для його обчислення необхідно отримати перетин та об’єднання слів двох документів та поділити їх. Це виражається формулою:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Для обчислення коефіцієнта Жаккара необхідно спочатку провести лемитизацію текстів для перетворення всіх важливих слів до кореневої форми для кращої точності показника.

Наприклад, маємо дві фейкові новини нібито про те, що фінансування патріотичного кіно збільшать за рахунок фонду боротьби з коронавірусом[31] (рисунок 2.2 та 2.3).

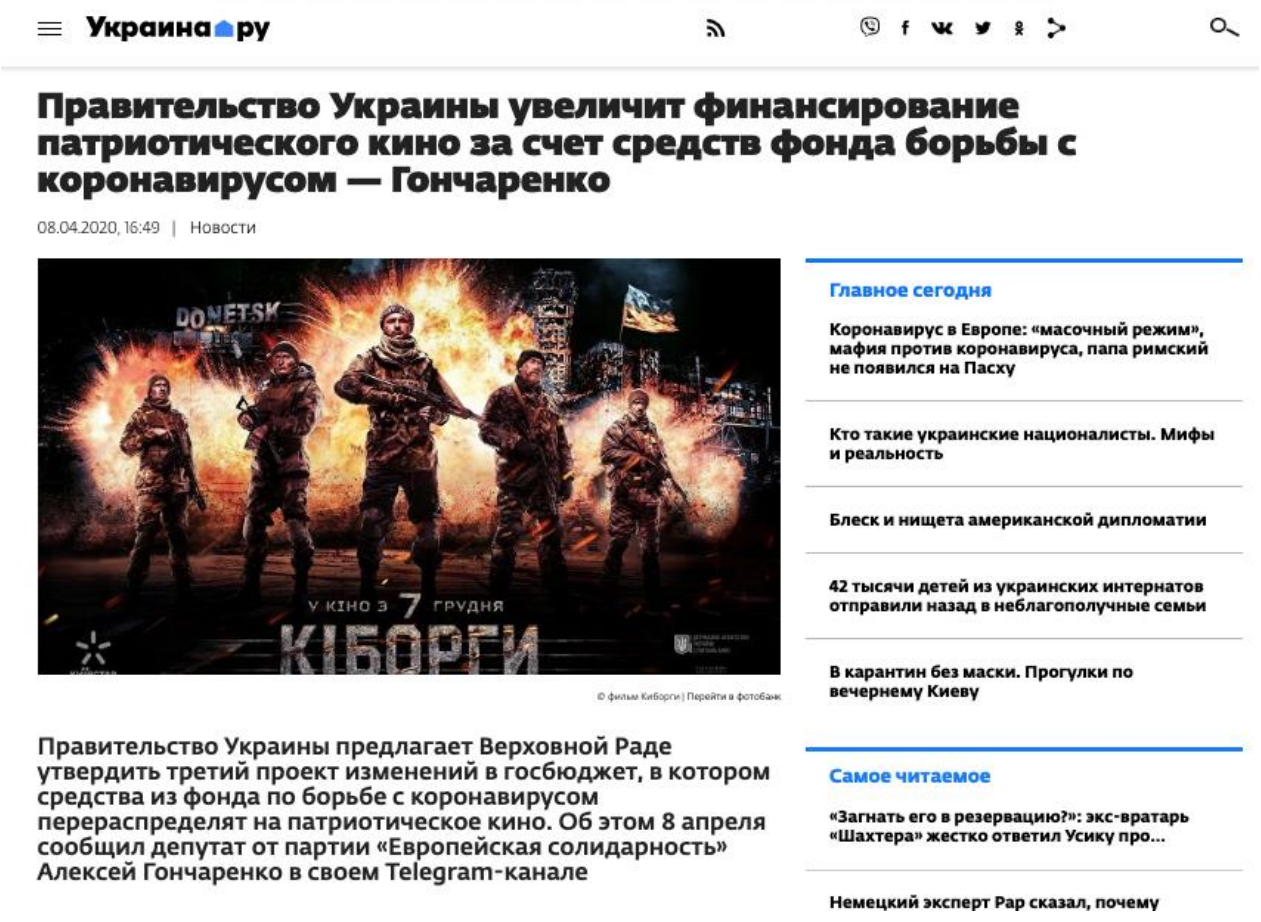

Рисунок 2.2 – Фейкова новина на порталі Украина.ру

Рисунок 2.3 – Фейкова новина на порталі Strana.ua

Обчислимо коефіцієнт Жаккара для заголовків двох новин.

Заголовок 1: «Правивительство Украины увеличит финансирование патриотического кино за счет средств фонда борьбы с коронавирусом».

Заголовок 2: «Забрать у фонда по борьбе с коронавирусом и отдать на патриотическое кино».

Після проведеня лематизації та видалення шуму отримаємо набір слів для заголовків:

Заголовок 1: Правительство, Украина, увеличить, финансирование, патриотический, кино, счет, фонд, борьба, коронавирус.

Заголовок 2: Забрать, фонд, борьба, коронавирус, отдать, патриотический, кино.

Побудуємо діаграму Венна для двох заголовків (рисунок 2.3).

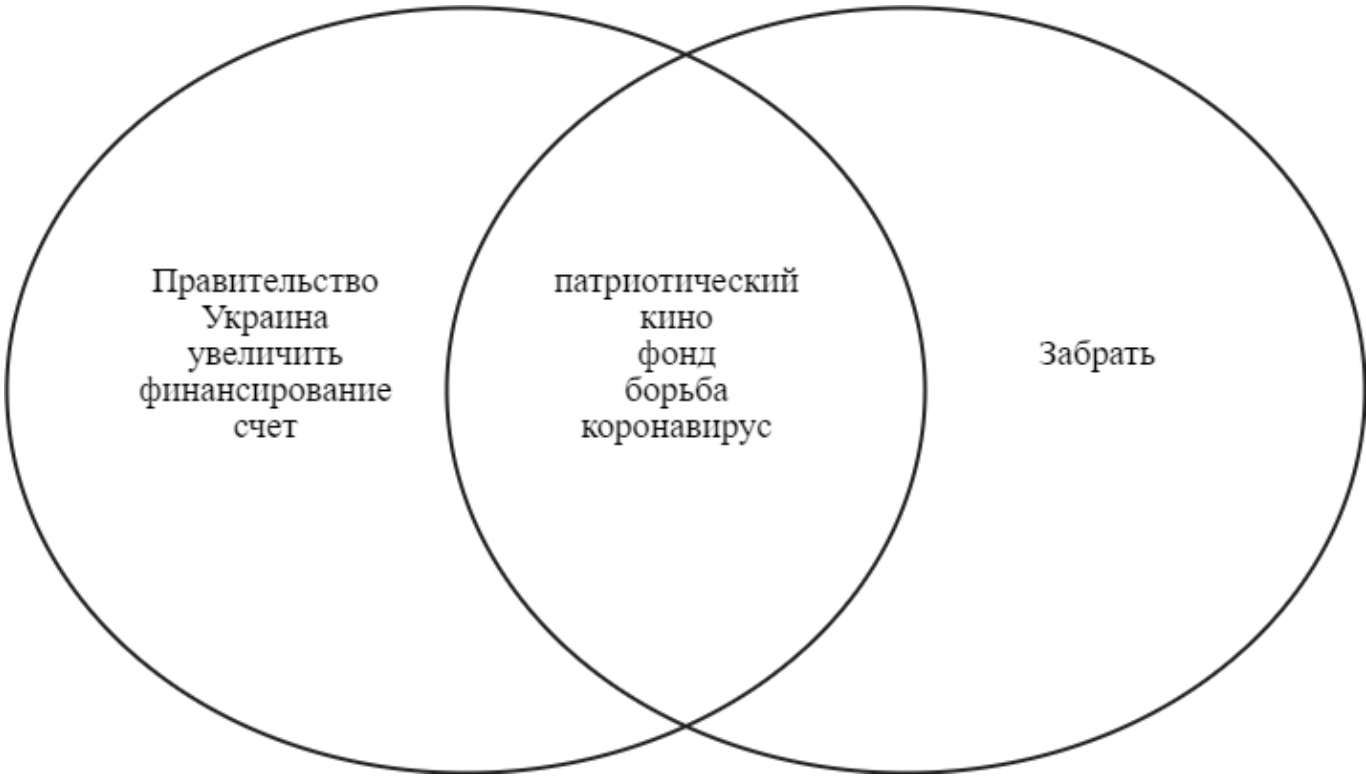

Рисунок 2.4 – Діаграма Венна для двох заголовків

Обчислимо коефіцієнт Жаккара:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{5}{5+5+1} = 0.45$$

Можна зробити висновок, що дві новини на одну тематику.

Іншим показником подібності є косинус подібності[32]. Косинус подібності – це міра подібності двох ненульових векторів внутрішнього простору, що вимірюється як косинус кута між ними. Косинус 0° дорівнює 1 і менше 1 для будь-якого кута в інтервалі (0, π] радіанів. Для його обчислення використовується формула:

$$similarity = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

де A_i і B_i- компоненти векторів A та B відповідно.

Для обчислення косинусу подібності двох векторів нас необхідно перетворити речення фейкових новин у векторний вигляд. Для цього можна використати мішок слів та TF-IDF, описані вище. Вибір TF-IDF залежить від застосування і не залежить від того, як насправді обчислюється косинус подібності. Посилаючись на дослідження[13], обчислення косинуса подібності в термінах TF-IDF перетворюється на:

$$\cos(\theta) = \frac{TFIDF_{news1} \cdot TFIDF_{news2}}{|TFIDF_{news1}||TFIDF_{news2}|}$$

Обчислимо косинус подібності для новин, розглянутих вище. Побудуємо таблицю Bag of words для двох заголовків, тобто кількість зустрічі кожного слова в тексті (таблиця 2.1).

Таблиця 2.1 – Bag of words двох заголовків

	1	2
Правительство	1	0
Украина	1	0
увеличить	1	0
финансирование	1	0
патриотический	1	1
кино	1	1
счет	1	0
фонд	1	1
борьба	1	1
коронавирус	1	1
забрать	0	1

Нормалізуємо TF для двох заголовків (Таблиця 2.2).

Таблиця 2.2 – Нормалізований TF двох заголоків

	1	2	TF-IDF 1	TF-IDF 2
Правительство	1	0	0,090909091	0
Украина	1	0	0,090909091	0
увеличить	1	0	0,090909091	0
финансирование	1	0	0,090909091	0
патриотический	1	1	0,090909091	0,090909091
кино	1	1	0,090909091	0,090909091
счет	1	0	0,090909091	0
фонд	1	1	0,090909091	0,090909091
борьба	1	1	0,090909091	0,090909091
коронавирус	1	1	0,090909091	0,090909091
забрать	0	1	0	0,090909091

Обчислимо IDF для кожного заголовка (Таблиця 2.3):

Таблиця 2.3– IDF двох заголовків

	1	2	Normalized TF 1	Normalized TF 2	IDF 1	IDF 2
Правительство	1	0	0,090909091	0	1,651019245	1,651019245
Украина	1	0	0,090909091	0	1,651019245	1,651019245
увеличить	1	0	0,090909091	0	1,651019245	1,651019245

Продовження таблиці 2.3

финансирование	1	0	0,090909091	0	1,651019245	1,65101925
патриотический	1	1	0,090909091	0,090909091	1	1
кино	1	1	0,090909091	0,090909091	1	1
счет	1	0	0,090909091	0	1,651019245	1,65101925
фонд	1	1	0,090909091	0,090909091	1	1
борьба	1	1	0,090909091	0,090909091	1	1
коронавирус	1	1	0,090909091	0,090909091	1	1
забрать	0	1	0	0,090909091	1,651019245	1,65101925

Отримаємо TF-IDF для кожного заголовка (Таблиця 2.4)

Таблиця 2.4– IDF двох заголовків

IDF 2	TF-IDF 1
1,65101925	0,150093
1,65101925	0,150093
1,65101925	0,150093
1,65101925	0,150093
1	0,090909

Продовження таблиці 2.4

1	0,090909
1,65101925	0,150093
1	0,090909
1	0,090909
1	0,090909
1,65101925	0

Обчислимо косинус подібності для двох заголовків фейкових новин:

$$\cos(\theta) = \frac{TFIDF_{news1} \cdot TFIDF_{news2}}{|TFIDF_{news1}||TFIDF_{news2}|} = \frac{0{,}041}{0{,}39 * 0{,}25} = 0{,}41$$

Отримана схожість двох заголовків з вірогідністю 41%.

Виконаємо порівняння ефективності коефіцієнта Жаккара та косинуса подібності[33].

– Коефіцієнт Жаккара приймає лише унікальний набір слів для кожного речення / документа, тоді як косинус подібності приймає загальну довжину векторів. (ці вектори можна отримати за допомогою Bag of Words та TF-IDF).

– Це означає, що якщо ви повторите певне слово у одному із текстів кілька разів, подібність косинусів змінюється, але коефіцієнт Жаккара.

– Коефіцієнт Жаккара хороший для випадків, коли дублювання не має значення, косинус подібності хороша для випадків, коли дублювання має значення під час аналізу подібності тексту.

На основі порівняння було вирішено використовувати косинус подібності, оскільки на великому обсязі даних повторення слів відіграє важливу роль. Також у фейкових новинах певні події, особистості та факти повторюється, тому це ще одна перевага для косинуса подібності.

2.4 Побудова графу розповсюдження фейкової новини

Основною метою наукової роботи є дослідження розповсюдження фейкових новин. На основі розгянутої математичної моделі створений алгоритм пошуку подібних фейкових новин за допомогою косинусу подібності та часових міток.

Наведемо псевдокод та алгоритм (рис.2.5) знаходження подібності новин та генерування графу розповсюдження дезінформації.

```
queryNews = 'text'
queryNewsVector = tfidf(queryNews.text)
similarNews = []
for news in newsArray:
    currentNewsVector = tfidf(news.text)
    if similarity(queryNewsVector, currentNewsVector) > 0.6:
        similarNews.append(news)
similarNews.sort(key=lambda news: news.date, reverse=True)
generateGraph()
```

Рисунок 2.5 – Алгоритм генерування графу розповсюдження дезінформації

Розглянемо на приладі фейкової новини про те, що в Україні за допомогу у висаджені городів штрафують на 17 тис грн (рисунок 2.3)[34].

Рисунок 2.3 – Фейкова новина на порталі Agronews за 11 квітня 2020 року

Цю новину було опубліковано 11 квітня 2020 року о 16:14. За допомогою косинуса подібності було знайдено подібні новини, які були опубліковані

після цього [35], [36] (рисунки 2.4 та 2.5).

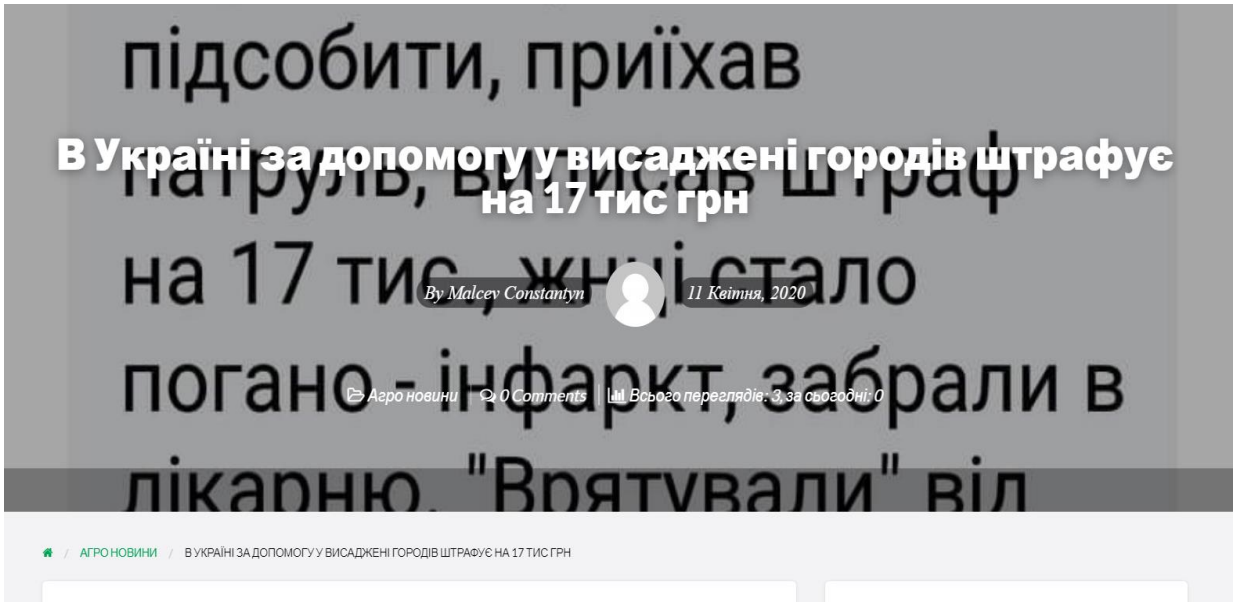

Рисунок 2.4 – Фейкова новина на порталі Юнікорнс за 11 квітня 2020 року

Рисунок 2.5 – Фейкова новина на порталі ВолиньІнфо за 12 квітня 2020 року

Отримали три новини з часовими мітками та косинусами подібності (Таблиця 2.5):

Таблиця 2.5 – Порівняння фейкових новин

Новина	Часова мітка	Косинус подібності відносно першої
[34]	11 квітня 2020 року о 16:14	-
[35]	11 квітня 2020 року о 20:05	1
[36]	12 квітня 2020 року о 12:04	0,8

На основі отриманих даних будується граф з шляхом розповсюдження фейкової новини з наданням інформації про час публікації та схожості фейкової новини з першою опублікованою (Рисунок 2.6)

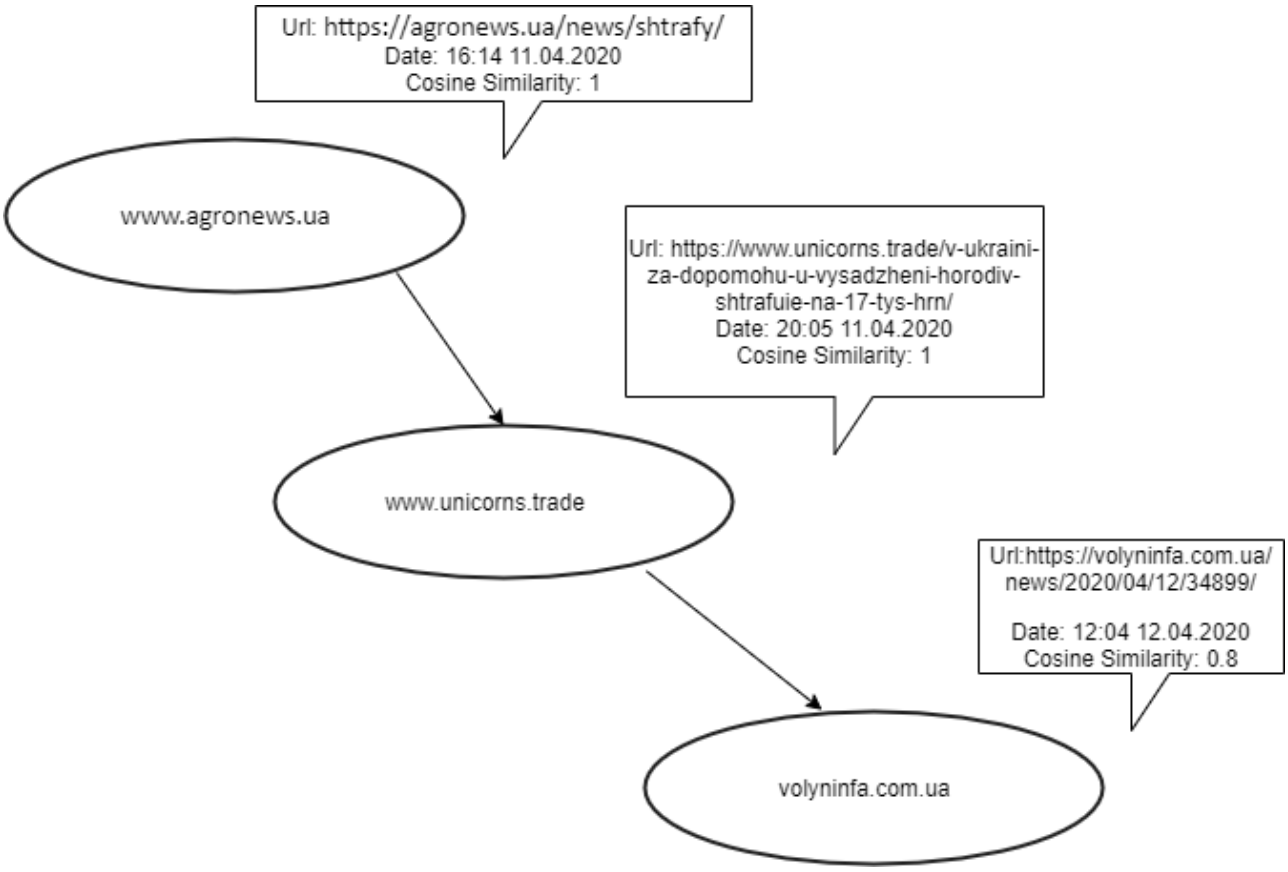

Рисунок 2.6 – Граф розповсюдження фейкової новини

Загальний алгоритм відображений на рисунку 2.7.

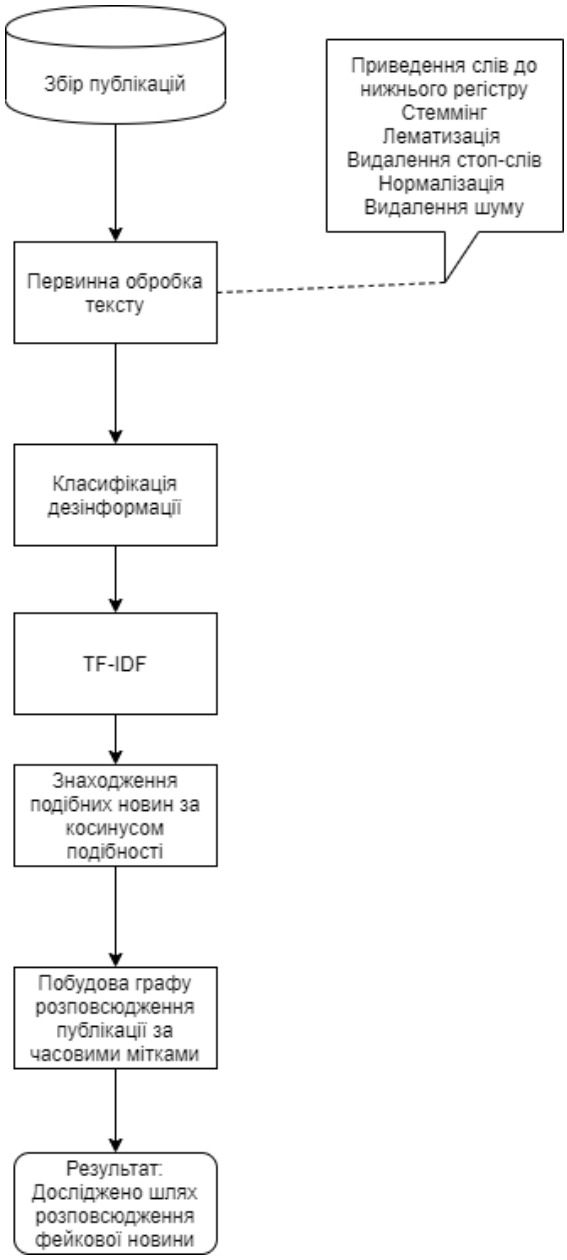

Рисунок 2.7 – Загальний алгоритм

2.4 Висновки до розділу

Пошук та знаходження фейкових новин – це дуже важлива задача в сучасному світі. У зв’язку із війною на Сході України та поширення епідемії коронавірусу ця задача стала особливо важливою. На сьогодні автоматичних способів визначення шляху розповсюдження фейкової новини для української та російської мови не існує.

Тому задачу наукової роботи можна сформулювати як дослідження способу знаходження першоджерела фейкової новини та генерація графу розповсюдження фейкової новини. Пропонується використання часових міток публікації новин та використання косинуса подібності для знаходження подібних фейкових новин. Для вирішення цієї задачі необхідно виконати такі задачі: згенерувати актуальний датасет з новинами, знайти схожі новини за допомогою косинуса подібності, побудувати граф розповсюдження фейкової новини на основі часових міток.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ГЕНЕРУВАННЯ ШЛЯХУ РОЗПОВСЮДЖЕННЯ ДЕЗІНФОРМАЦІЇ

3.1 Вимоги до розроблюваного програмного забезпечення

Відповідно до завдання цієї роботи та аналізу існуючого програмного забезпечення., для реалізації генерування шляху розповсюдження дезінформації необхідно:

– Розробити класифікатор елементів дезінформації та генератор шляху розповсюдження дезінформації з підтримкою української та російської мови.

– Імплементувати запропоновані алгоритми: алгоритми первинної обробки тексту, алгоритм аналізу тексту за сентиментом, обчислення метрик TF-IDF, обчислення косинусу подібності, побудова графу розповсюдження текстової новини.

– Розробити інтерфейс користувача для користування програмним забезпеченням.

– Протестувати алгоритм генерування шляху розповсюдження дезінформації на різних новинах.

3.2 Архітектура ПЗ

На основі підходів з попереднього розділу сформуємо архітектуру програмного забезпечення. Для опису архітектури розроблено такі типи UML-діаграм:

– Діаграма класів.

– Діаграми послідовностей.

– Діаграма розгортання.

Перед початком розробки діаграм необхідно виділити основні архітектурні рішення, які були використані під час розробки.

Основною задачею при розробці програмного забезпечення була підтримка україномовних та російськомовних текстів. Для первинної обробки

тексту було використано морфологічний аналізатор pymorphy2[37]. Для семантичного аналізу російських текстів була використана бібліотека Dostoevsky[38]. Ці бібліотеки написані на мові програмування Python[39], тому для розробки програмного забезпечення було вирішено використовувати саме її. Веб-інтерфейс розроблений за допомогою TypeScript фреймворку Angular[40]. Для стилізації веб-інтерфейсу використано набір компонентів Angular Material[41].

Детально опишемо основні етапи розробки програмного забезпечення та технології, які використовуються для їх реалізації.

Для покращення результатів класифікації новин на наявність елементів дезінформації проводиться первинна обробка тексту. Вона включає такі етапи:

– Приведення слів до нижнього регістру;
– Стеммінг;
– Лематизація;
– Видалення стоп-слів;
– Нормалізація;
– Видалення шуму;

Для приведення слів до нормальної форма використовується бібліотека pymorphy2. Pymorphy2 – це морфологічний аналізатор, написаний на мові python. Ця бібліотека має такий функціонал:

– Приводить слова до нормальної форми.
– Ставить слова в правильну форму. Наприклад, ставить слово в множині, змінює відмінок слова.
– Повертає граматичну інформацію про слово (число, рід, відмінок, частина мови).

Особливістю бібліотеки є підтримка української та російської мови. При роботі бібліотеки використовується словник OpenCorpora[42].

Для незнайомих слів будуються гіпотези. Pymorphy2 використовує такі механізми для будування гіпотез:

– Відсікання невідомих префіксів: у pymorphy2 для кожної підтримуваної мови зберігається невеликий список таких префіксів (наприклад, "не", "анти", "псевдо", "супер", "дво" і т.д. для російського). Якщо слово починається з одного з таких префіксів, то pymorphy2 відсікає префікс, розбирає те, що залишилося, а потім приписує префікс назад.

– Відсікання відомих префіксів: якщо 2 слова відрізняються тільки тим, що до одного з них щось приписано спереду, то, швидше за все, і схилятися вони будуть однаково. Тому якщо розбирається слово можна уявити як ПРЕФІКС + якесь інше слово зі словника, то pymorphy2 вважає, що слово розбирається так само, як і це інше слово.

– Передбачення по кінця слова

– Прислівники на по-: знайдене слова передбачаються як прислівники, якщо вірно наступне:

– слово починається на "по-";

– воно довше 5 символів;

– частина слова без "по-" може бути розібрана як прикметник однини давального відмінка.

– Частка, відокремлена дефізом: іноді зручно слова, до яких через дефіс приписаний дефіс, розбирати як єдиний токен. Тому pymorphy2 вміє розбирати токени на кшталт «дивись-но» або «подивився-таки».

– Складові слова, записані через дефіс: pymorphy2 підтримує складові слова з двох частин, розділених дефісом. Для таких слів pymorphy2 спочатку розбирає обидві частини окремо (вони можуть бути несловниковими словами).

– Ініціали: токени з однією буквою в верхньому регістрі pymorphy2 прогнозує як ініціали: для них повертаються варіанти розбору "ім'я" і "батькові", по всіх родів, відмінками і числам.

– Сортування результатів розбору: при прогнозі по кінцю слова результати упорядковано відповідно до "продуктивності" варіантів розбору: найбільш продуктивні варіанти будуть першими.

Для семантичного аналізу російської мови використовується бібліотека Dostoevsky, написана на мові Python. Модель для аналізу була натренована на RuSentiment датасет [43] і досягла результатів F1= ~0.71. RuSentiment – це новий набір даних для аналізу настроїв повідомлень в соціальних мережах російською мовою. Наразі RuSentiment є найбільшим у своєму класі для російської мови, де є 30521 допис зазначається капсою Флісса з значенням 0,58 (3 анотації на допис). Для диверсифікації набору даних попередньо було обрано 6 749 публікацій за допомогою активної стратегії навчання.

Для семантичного аналізу української мови використовується модель розроблена в рамках наукової роботи[44], який показав гарні результати на тестових даних.

Програмний застосунок розроблений на операційній системі Windows 10 та на мові програмування Python 3.8.

Діаграма класів для розробленої системи зображена на рисунку 3.1.

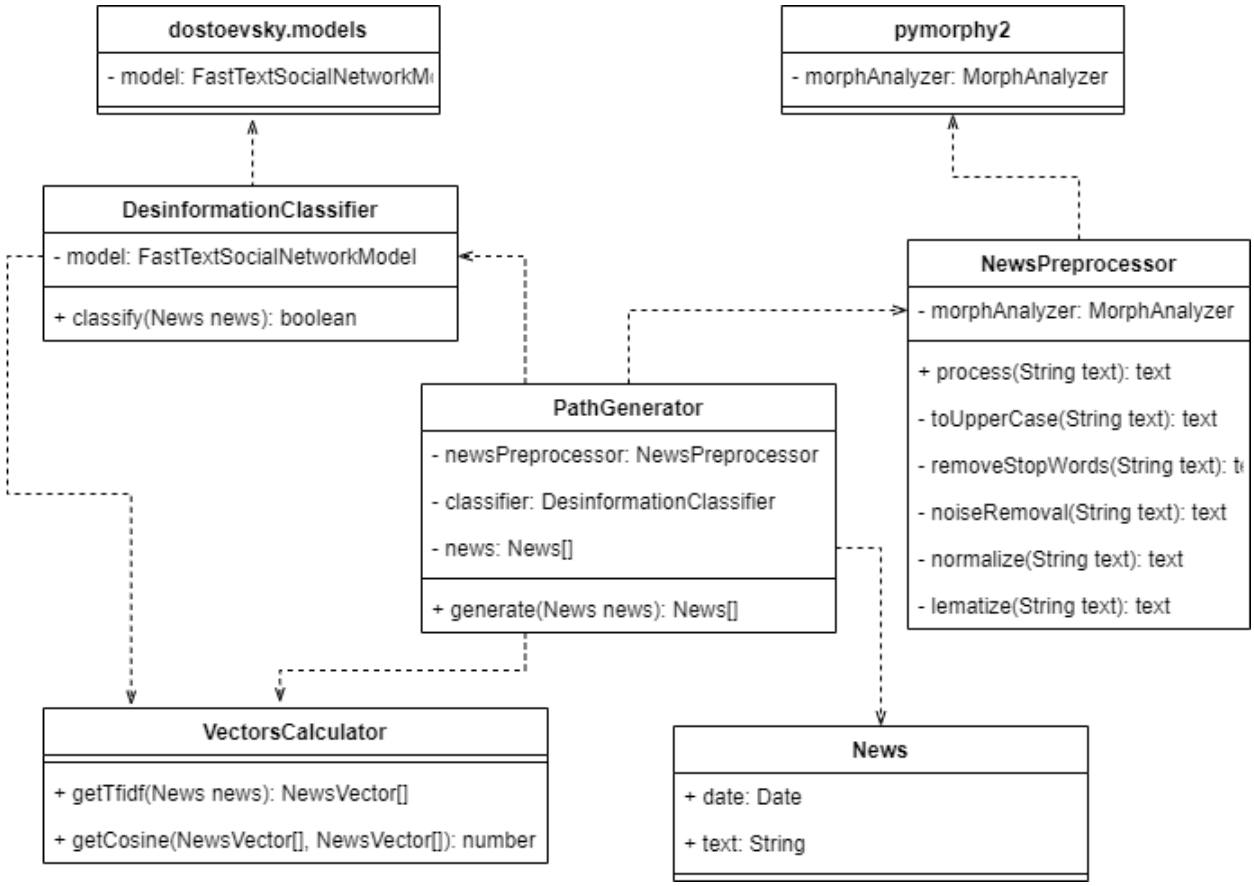

Рисунок 3.1 – UML-діаграма класів

Розпишемо функціонал кожного класу:

– NewsPreprocessor: відповідає за первинну обробку тексту. Використовує MorphAnalyzer бібліотеки pymorph2. Опис методів:

– toUpperCase: приведення слів до нижнього регістру;

– lemmatize: лематизація;

– removeStopWords: видалення стоп-слів;

– normalize: нормалізація;

– noizeRemoval: видалення шуму;

– DesinformationClassifier: класифікація тексту за сентиментом. Використовує модель FastTextSocialNetworkModel бібліотеки Dostoevsky. Опис методів:

– Classify: виконує класифікацію конкретного тексту за сентиментом.

– VectorsCalculator: реалізація алгоритмів для обчислення TF-IDF та косинуса подібності двох новин. Опис методів:

– getTfidf: перетворює текст новини в векторний вигляд;

– getCosine: обчислює косинус подібності двох новин;

– PathGenerator: основний клас для генерації шляху розповсюдження дезінформації. Реалізує основний алгоритм програмного застосунку. Метод generate повертає відсортований масив новин, який показує шлях розповсюдження дезінформації.

Наведемо діаграму послідовностей для таких процесів:

– Класифікація тексту на наявність елементів дезінформації.

– Генерація шляху розповсюдження фейкової новини.

Діаграма послідовності для класифікації тексту наведена на рисунку 3.2.

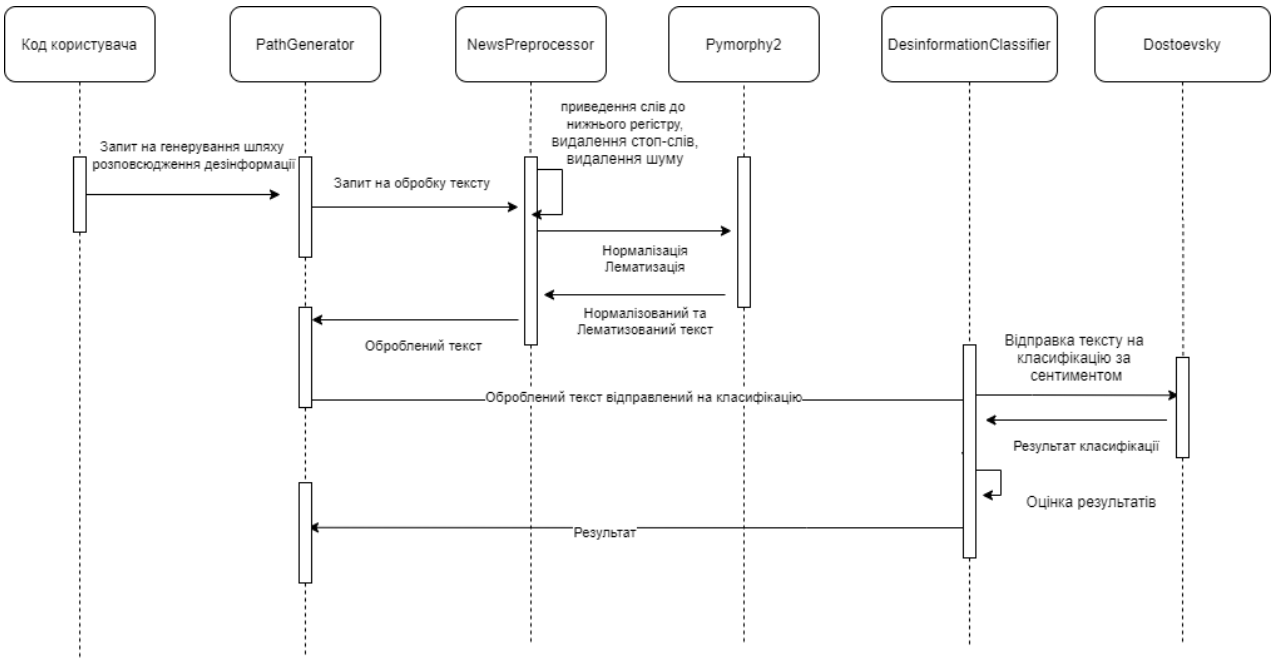

Рисунок 3.2 – UML-діаграма послідовностей класифікації тексту на наявність елементів дезінформації

Користувач відправляє запит на генерування шляху розповсюдження дезінформації. PathGenerator отримає вхідний текст та відправляє його на

обробку в NewsPreprocessor. Методи NewsPreprocessor виконують первинну обробку тексту. Також за допомогою бібліотеки Pymorphy2 відбувається нормалізація та лематизація тексту. Оброблений текст повертається в PathGenerator та передається в DesinformationClassifier для подальшої класифікації. DesinformationClassifier за допомогою бібліотеки Dostoevsky класифікує текст за сентиментом та повертає результати класифікації в DesinformationClassifier. DesinformationClassifier оцінює результати на негативне забарвлення та повертає результат в PathGenerator.

Діаграма послідовностей для генерації шляху розповсюдження фейкової новини зображено на рисунку 3.3.

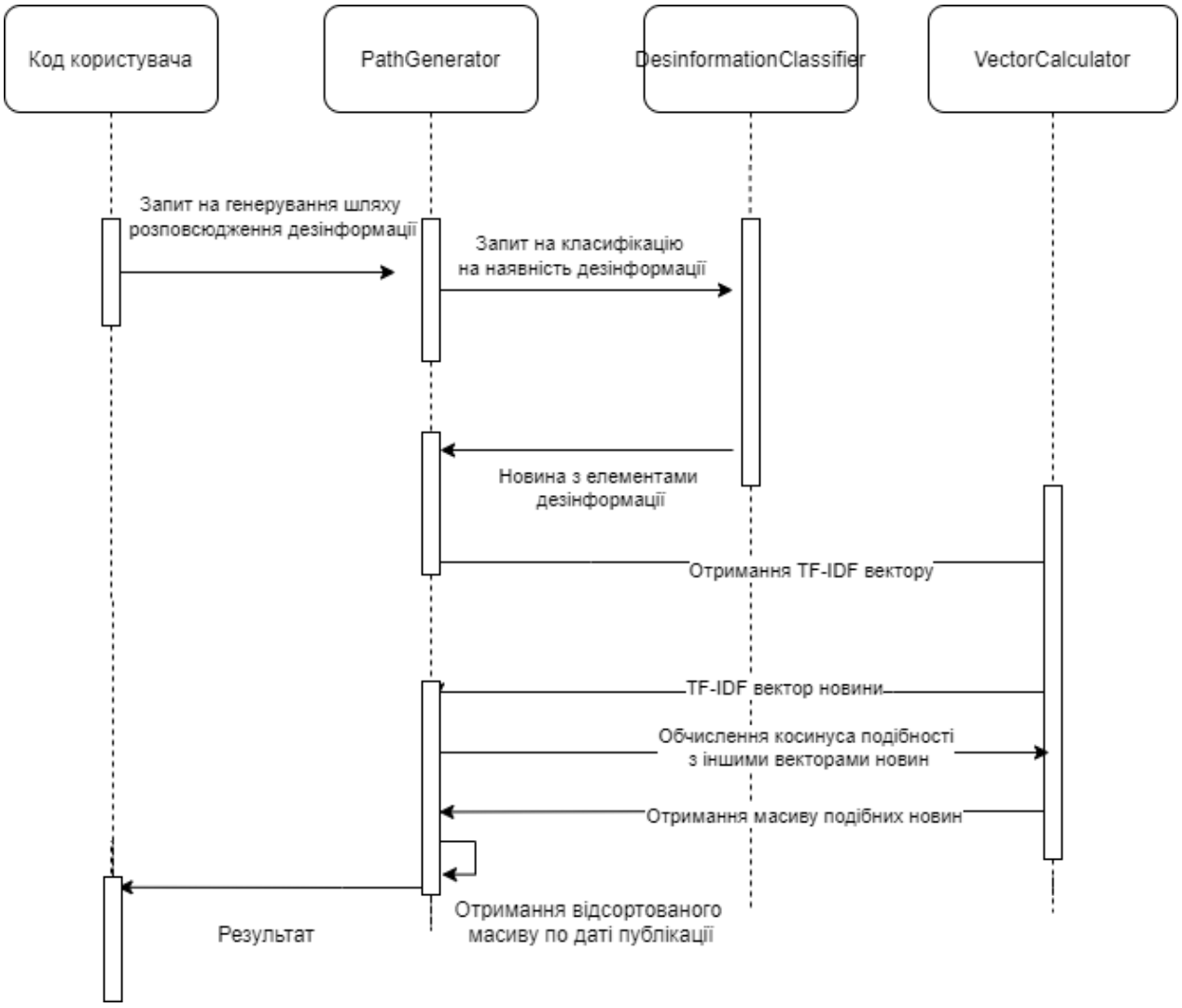

Рисунок 3.3 – UML-діаграма послідовності генерації шляху розповсюдження дезінформації

Користувач подає на вхід текст новини з датою її публікації. PathGenerator відправляє текст новини на класифікацію на наявність елементів дезінформації за допомогою класу DesinformationClassifier. Якщо були знайдені елементи дезінформації в вхідній новині, то обчислюється TF-IDF вектор для вхідної новини за допомогою методу getTfidf класу VectorCalculator. Далі відбувається перевірка з базою новин і обчислюються косинуси подібності. На виході отримуємо відсортований масив масив подібних новин відносно дати публікації новини. На основі отриманого масиву генерується граф розповсюдження дезінформації.

Діаграму розгортання зображено на рисунку 3.4.

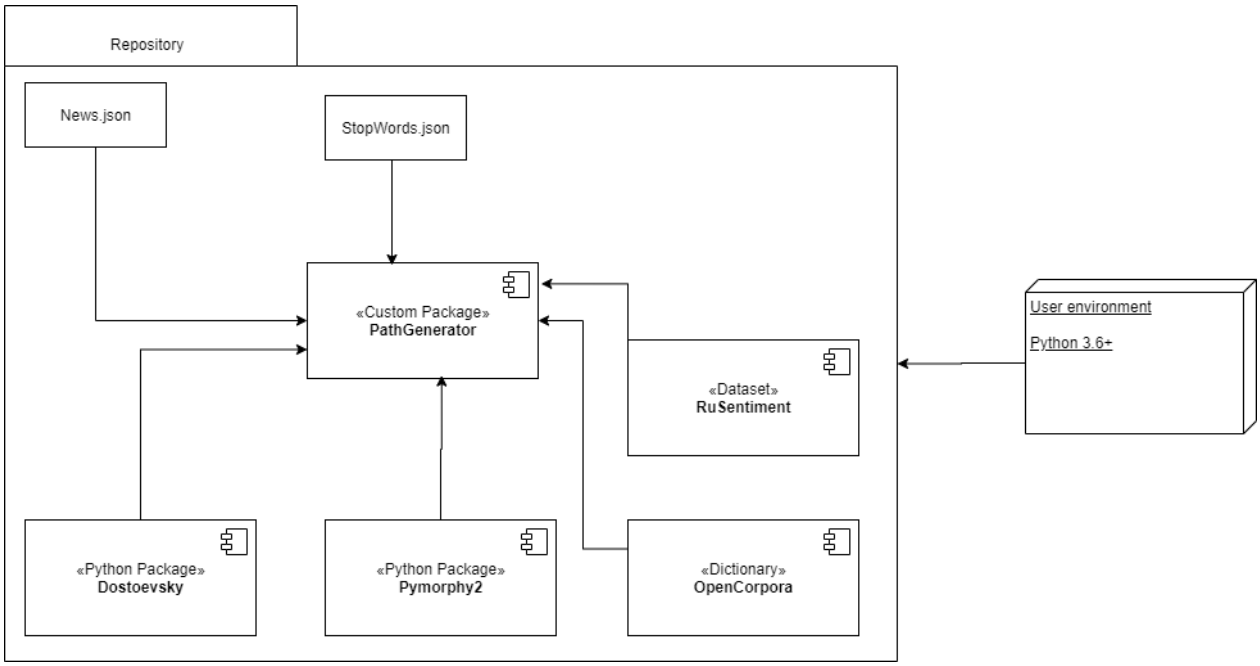

Рисунок 3.4 – UML-діаграма розгортання

Для успішного розгортання користувач має встановити Python 3.6+, встановити необхідні пакети та завантажити словники.

3.3 Керівництво користувача

Для успішно встановлення та користування застосунком надано керівництво користувача. Наведено встановлення програмного забезпечення та користування веб-інтерфейсом.

3.3.1 Встановлення застосунку

Для коректної роботи програмного забезпечення необхідно встановити Python версії 3.6 і вище. Після розгортання програмного забезпечення необхідно встановити необхідні пакети за допомогою команди pip install.

Для коректного встановлення пакетів та словників необхідно також встановити Visual Studio Build Tools. Коректне встановлення пакетів протестовано на операційній системі Windows 10.

Для запуску серверу необхідно виконати команду py server.py

Для встановлення веб-інтерфейсу необхідно встановити NodeJs версії 12 і вище. Після цього необхідно виконати встановлення пакетів за допомогою команди npm install.

Для запуску веб-інтерфейсу необхідно виконати команду npm run start.

3.3.2 Використання застосунку

Розроблений застосунок представляє собою веб-інтерфейс, який дозволяє перевірити новину на наявність дезінформації та отримати граф розповсюдження дезінформації. Для локального використання користувачу необхідно запустити сервер, який приймає запити на обробку (рис. 3.5).

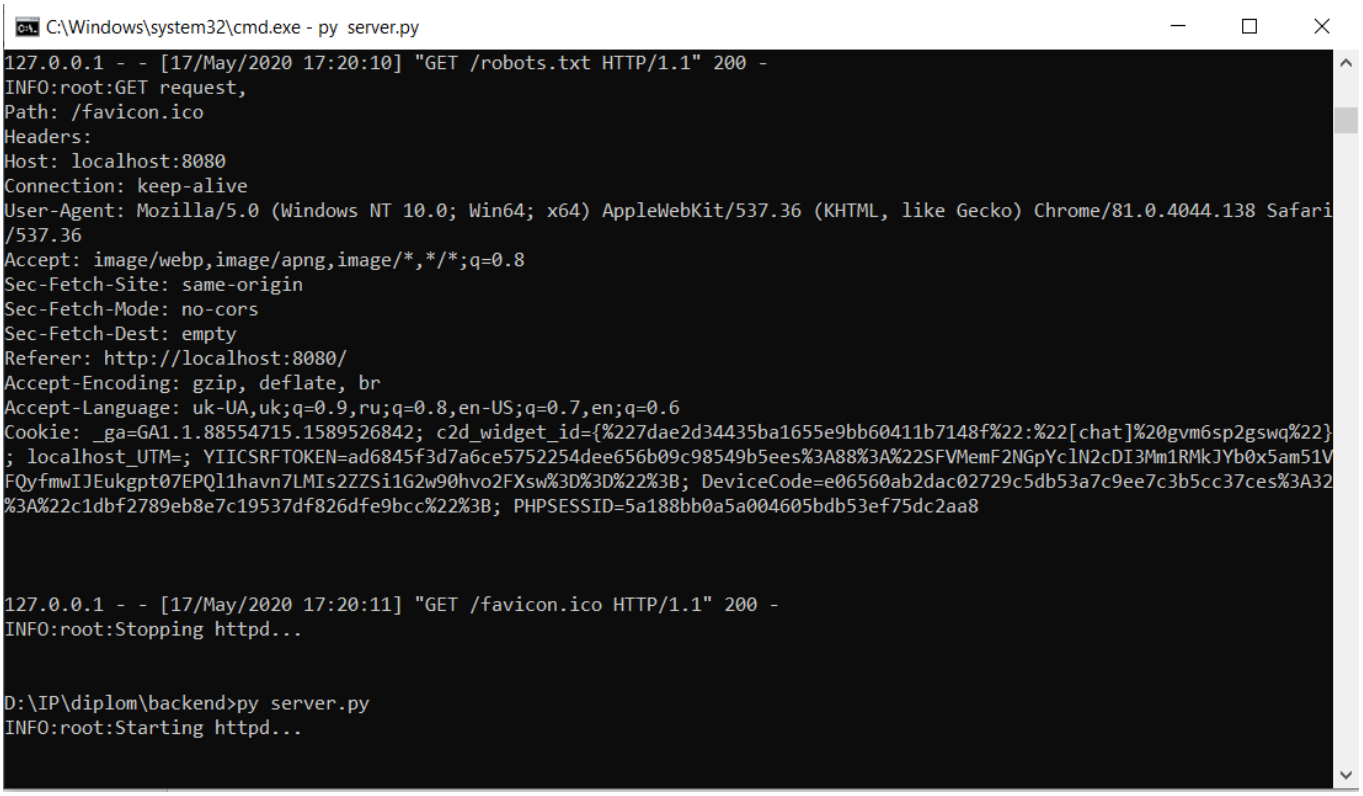

Рисунок 3.5 – Консольний вивід про запуск сервера

Далі користувачу необхідно відкрити веб-інтерфейс у веб-браузері на посиланням http://localhost:4200 (рис. 3.6).

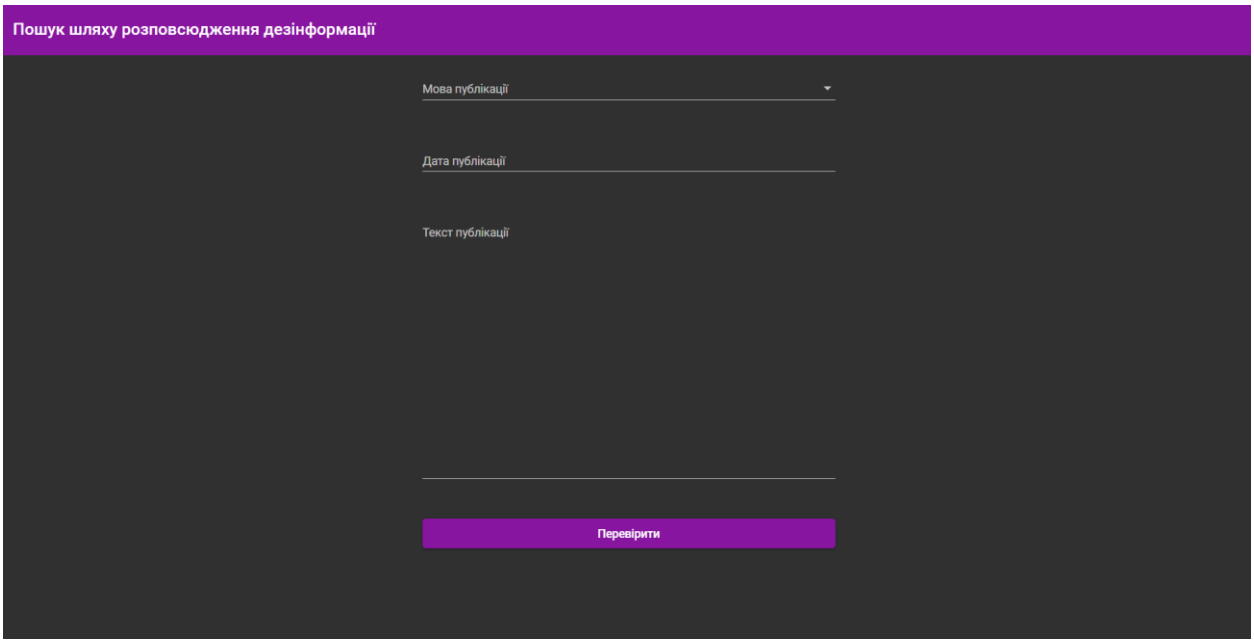

Рисунок 3.6 – Головна сторінка

Для перевірки тексту на наявність елементів дезінформації та генерацію шляху розповсюдження дезінформації користувачу необхідно ввести дату публікації та текст. Також необхідно ввести мову публікації. Для запуску перевірки необхідно натиснути кнопку «Перевірити» та дочекати результатів (рис.3.7).

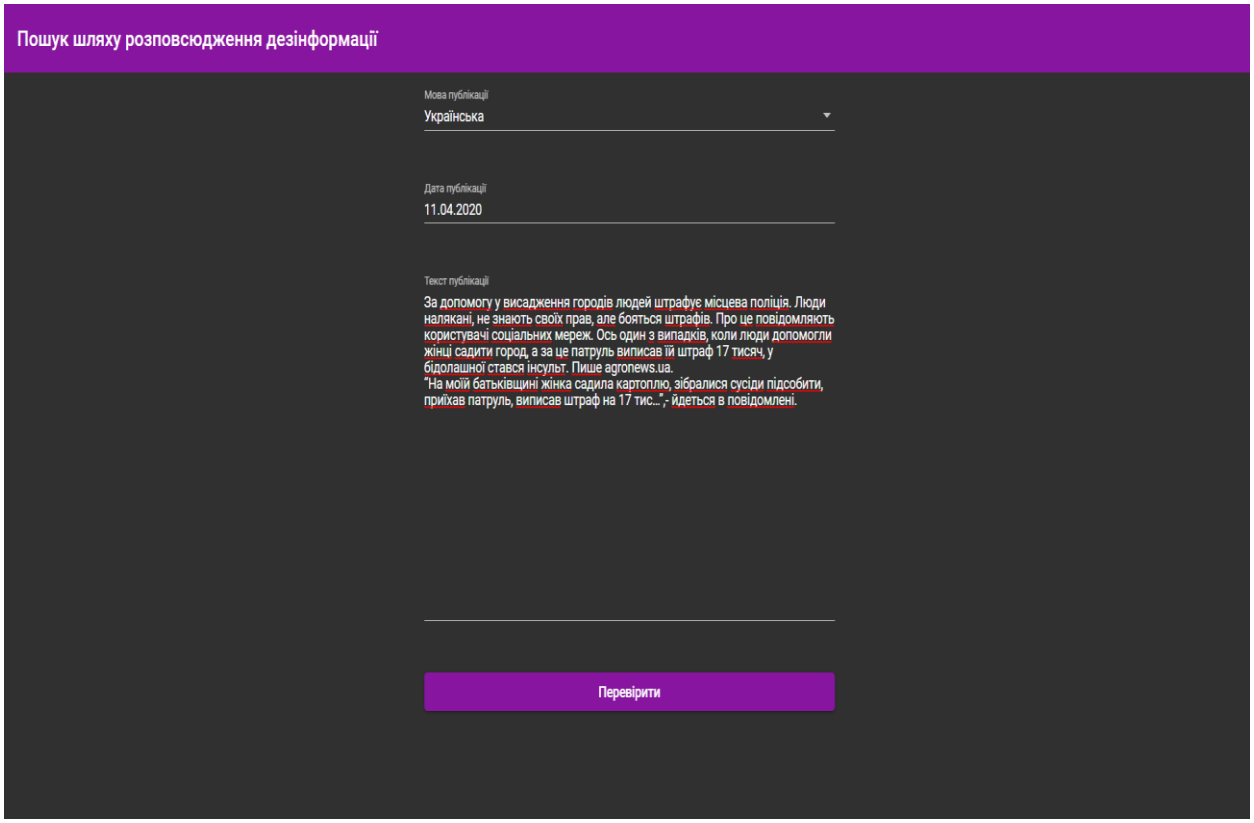

Рисунок 3.7 – Коректно введені дані для обробки

Результати обробки публікації продемонстровано на рисунку 3.8.

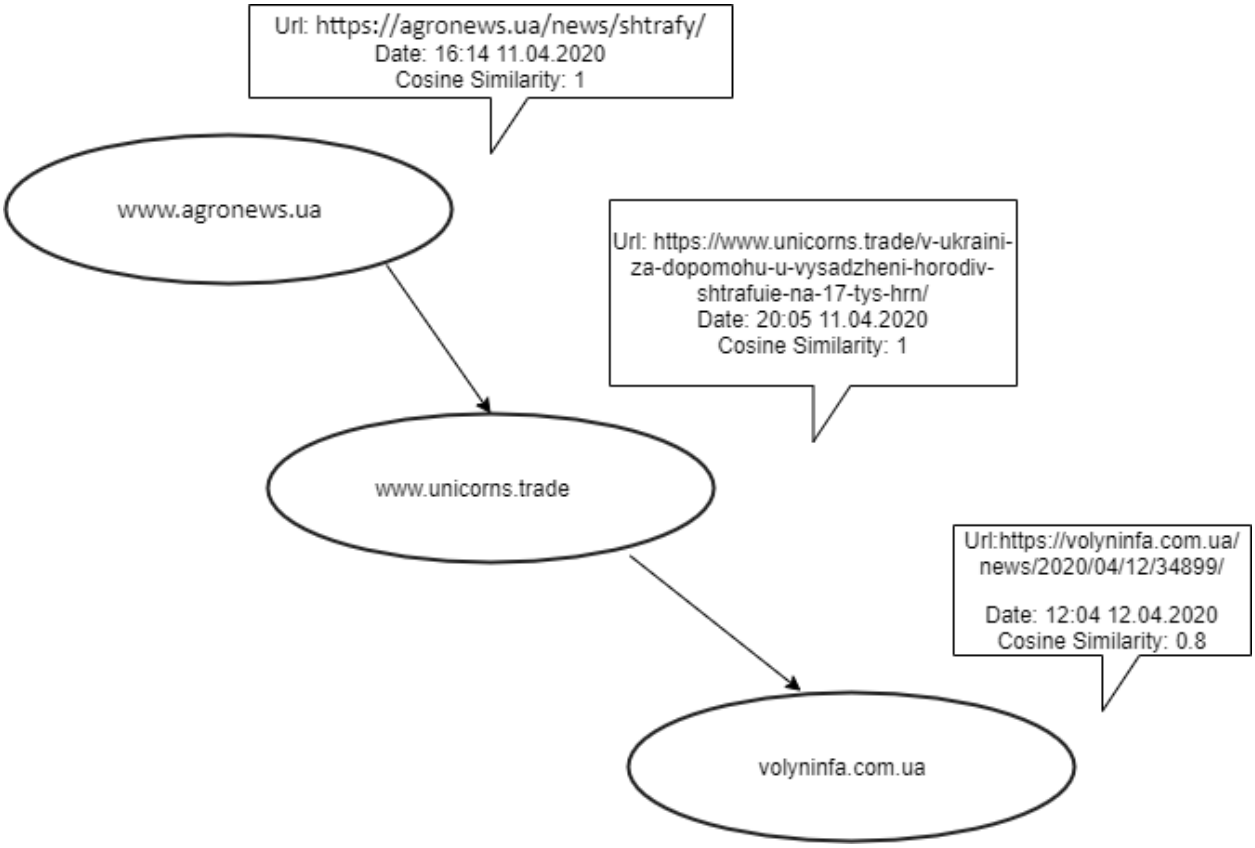

Рисунок 3.8 – Згенерований граф розповсюдження дезінформації

3.4 Висновки до розділу

У розділі була розглянута програмна реалізація методу виявлення елементів дезінформації та генерація графу розповсюдження дезінформації. Було розглянуто використані технологічні рішення, описана архітектура програмного забезпечення за допомогою UML-діаграм.

Було розроблено клієнт-серверний застосунок для генерації шляху розповсюдження дезінформації. Серверна частина розроблена за допомогою мови програмування Python. Для аналізу тексту було використано бібліотеку pymorphy2. Для класифікації тексту за сентиментом використано бібліотеку Dostoevsky. Доступна підтримка української та російської мови.

Веб-інтерфейс розроблений за допомогою TypeScript фреймворку Angular. Для стилізації веб-інтерфейсу використано набір компонентів Angular Material.

4 ОЦІНКА ЯКОСТІ РОБОТИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Під час розробки програмного забезпечення використовувалися алгоритми, які потребують оцінки якості. Була проведена оцінка якості класифікатора на згенерованому датасеті. Також було проведено порівняння обчислення косинуса подібності на різних даних з використанням мішка слів та TF-IDF. Також було проведено порівняння алгоритмів обчислення коефіцієнта Жаккара та косинуса подібності на різному наборі даних.

4.1 Оцінка якості класифікатора за сентиментом

Для оцінки якості роботи класифікатора його було запущена на різних алгоритмах. Результати тесту представлені в таблиці 4.1.

Продуктивність базового класифікатора зображена в таблиці 4.1.

Таблиця 4.1 – Продуктивність базового класифікатора

Класифікатор	F1	Точність	Recall
Logistic Regression	0.6884	0.6953	0.6946
Linear SVC	0.6856	0.6946	0.6925
Gradient Boosting	0.6848	0.6963	0.6919
NN classifier	0.7164	0.7199	0.7215

Графічне представлення зображено на рисунку 4.1.

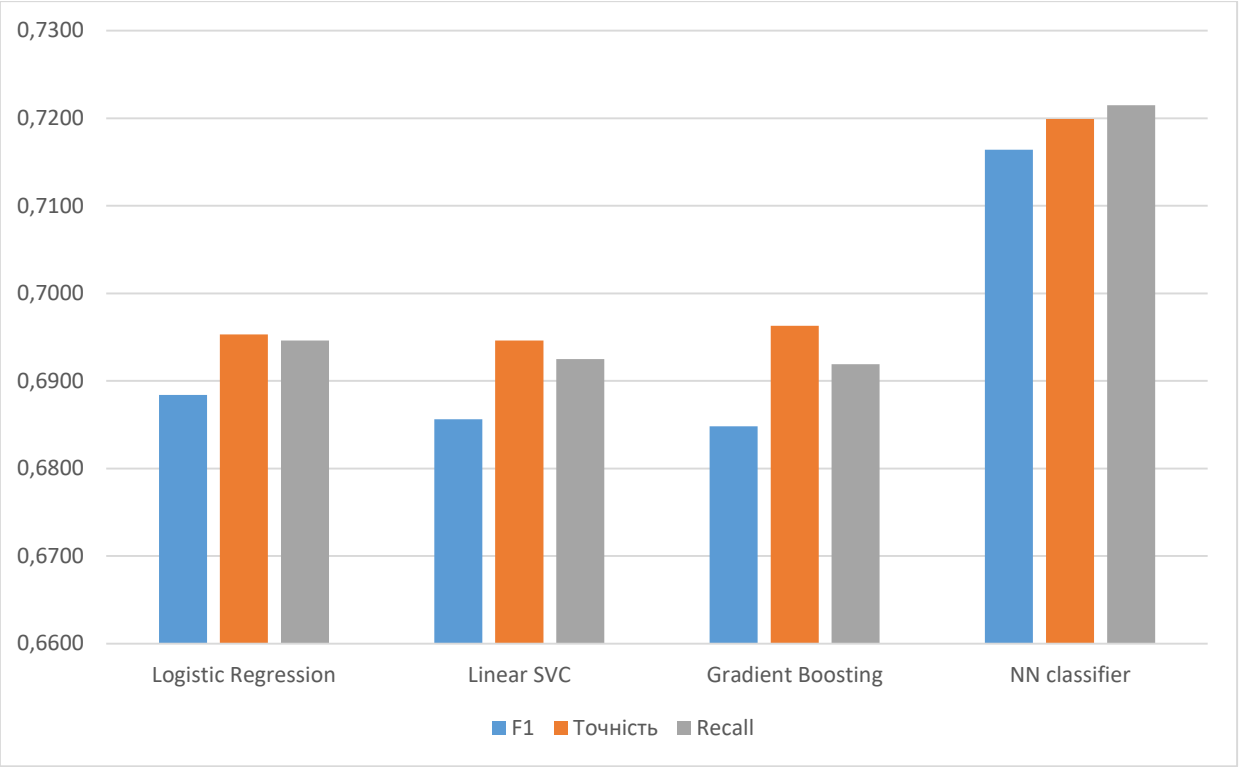

Рисунок 4.1 – Оцінка якості класифікатора за сентиментом

4.2 Порівняння мішка слів та TF-IDF при обчисленні косинуса подібності

Для доведення кращої точності TF-IDF на мішком слів в обчисленні косинуса подібності двох публікації було проведе тестові обчислення на різних типах тексту.

Отримані результати можна побачити на рисунку 4.2.

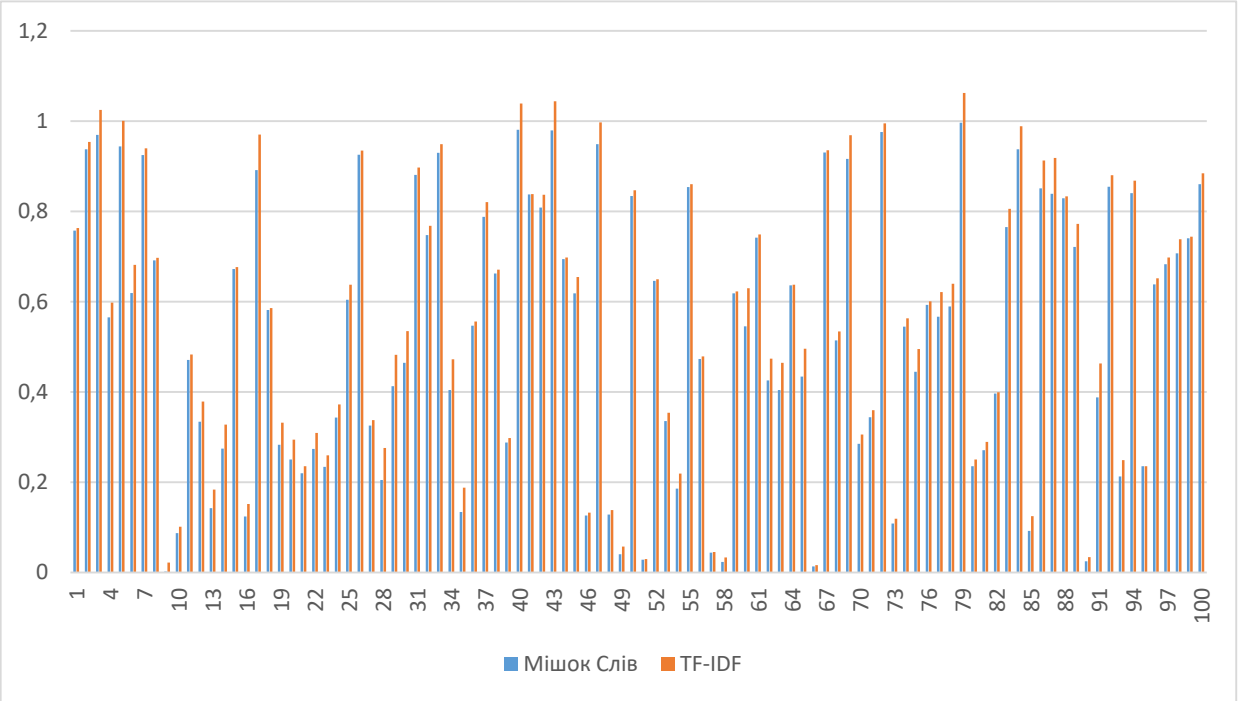

Рисунок 4.2 – Порівняння мішка слів та TF-IDF при обчисленні косинуса подібності

4.3 Порівняння точності результатів алгоритмів обчислення коефіцієнта Жаккара та косинуса подібності

При виборі алгоритма обчислення подібності двох публікацій були розглянуті дві міри схожості:

- Коефіцієнт Жаккара.
- Косинус подібності.

Провівши експеримент було вирішено використовувати косинус подібності. Результати порівняння зображенні на рисунку 4.3.

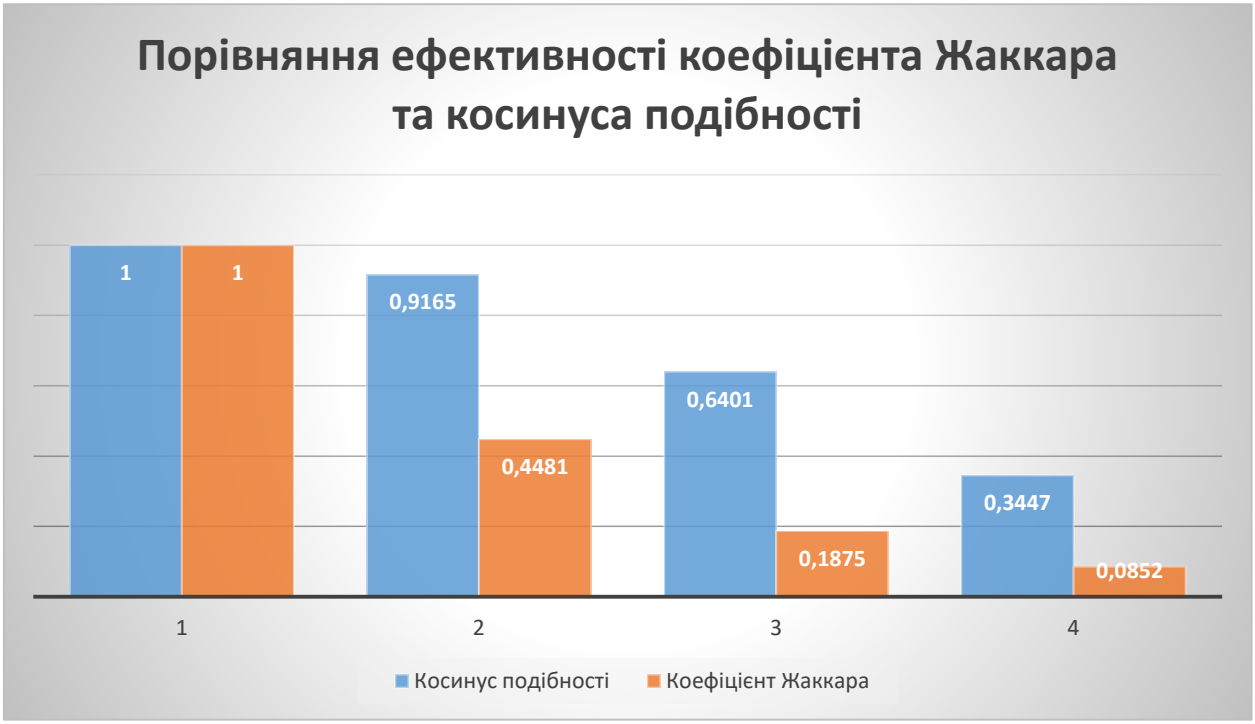

Рисунок 4.3 – Порівняння алгоритмів знаходження коефіцієнта Жаккара та косинуса подібності

4.4 Висновки до розділу

В цьому розділі наведено оцінку якості використаних алгоритмів, наведено порівняння з відомими аналогами.

В результаті використана модель для класифікації за сентиментом показала гарні результати (точність більше 0.69). Використання TF-IDF в заміну мішку слів та косинуса подібності призвело до покращення точності обчислення.

ВИСНОВКИ

У роботі був створений метод виявлення елементів дезінформації в потоці текстових даних та генерації шляху розповсюдження дезінформації.

У першому розділі було описано дезінформацію як явище, описано її класифікацію, шляхи та причини її створення, проаналізовано її розповсюдження. Було показано вплив її на світову політику та економіку. Розглянуто лінгвістичні та мануальні методи перевірки інформації на достовірність фактів. Портали перевірки інформації, такі як The Washington Post Fact Checker, Snopes, StopFake допомагають перевірити факти і встановити є новина правдивою. Описано шляхи поширення фейкових новин, мету їх створення.

У другому розділі було досліджено спосіб класифікації дезінформації, знаходження першоджерела фейкової новини та генерація графу розповсюдження фейкової новини. Було описано математичні моделі та алгоритми для досягнення цієї задачі. Для вирішення цієї задачі необхідно виконати такі задачі: згенерувати актуальний датасет з новинами, класифікувати новини на наявність дезінформації, знайти схожі новини за допомогою косинуса подібності, побудувати граф розповсюдження фейкової новини на основі часових міток.

У розділі програмної реалізації була розглянута програмна реалізація методу виявлення елементів дезінформації та генерація графу розповсюдження дезінформації. Було розглянуто використані технологічні рішення, описана архітектура програмного забезпечення за допомогою UML-діаграм.

Було розроблено клієнт-серверний застосунок для генерації шляху розповсюдження дезінформації. Серверна частина розроблена за допомогою мови програмування Python. Для аналізу тексту було використано бібліотеку pymorphy2. Для класифікації тексту за сентиментом використано бібліотеку

Dostoevsky та модель, розброблена в рамках наукової роботи[44]. Доступна підтримка україномовних та російськомовних текстів.

Веб-інтерфейс розроблений за допомогою TypeScript фреймворку Angular. Для стилізації веб-інтерфейсу використано набір компонентів Angular Material.

В розділі оцінки якості програмного забезпечення наведено оцінку якості використаних алгоритмів, наведено порівняння з відомими аналогами.

В результаті використана модель для класифікації елементів дезінформації має точність більше 0.69. Використання TF-IDF в заміну мішку слів та косинуса подібності призвело до покращення точності обчислення.

Науковою новизною є розробка методу виявлення елементів дезінформації в потоках даних з підтримкою обробки текстів української та російської мови .

ПЕРЕЛІК ПОСИЛАНЬ

1. This analysis shows how viral fake election news stories outperformed real news on Facebook. / [Craig Silverman]. // BuzzFeed News. – 2016.

2. Can fake news impact stock market. / [K. Rapoza]. . – 2017

3. Українці почали менше дізнаватися новини по телевізору, а більше – з соцмереж. [Електронний ресурс] – 2019. – Режим доступу до ресурсу: https://24tv.ua/ukrayintsi_diznayutsya_novini_u_2019_menshe_po_telefizoru_bilshe_z_sotsmerezh_infogarfika_n1222957

4. On deception and deception detection: Content analysis of computer-mediated stated beliefs. / [Victoria L Rubin] // – 2010 – С. 1–10

5. The validity effect: A search for mediating variables. Personality and Social Psychology Bulletin. / [Lawrence E Boehm] // – 1994 – С. 285–293

6. Confirmation bias: A ubiquitous phenomenon in many guises. Review of general psychology. / [Raymond S Nickerson] // – 1998– С. 175

7. Bandwagon, snob, and Veblen effects in the theory of consumers' demand. The quarterly journal of economics. / [Harvey Leibenstein] // – 1950– С. 183–207

8. Fake News Detection via NLP is Vulnerable to Adversarial Attacks. / [Zhixuan Zhou, Huankang Guan, Meghana Moorthy Bhat and Justin Hsu] // – 2017

9. A Review of Recent Advance in OnlineSpam Detection. / [Yingtong Dou] // – 2019

10. Probabilistic Context-Free Grammars (PCFGs). / [Michael Collins] // – 2019

11. The Language of Fake News: Opening the Black-Box of Deep Learning Based Detectors. / [Nicole O'Brien, Sophia Latessa, Georgios Evangelopoulos, Xavier Boix] // – 2019

12. Detecting Fake News with Sentiment Analysis and Network Metadata. / [Maniz Shrestha] // – 2019

13. Чотири рецепти дослідження фейків за допомогою онлайн-інструментів. [Електронний ресурс] – 2018. – Режим доступу до ресурсу: https://ms.detector.media/how-to/post/20527/2018-02-06-chotiri-retsepti-doslidzhennya-feikiv-za-dopomogoyu-onlain-instrumentiv/

14. Google Web Search. [Електронний ресурс] . – Режим доступу до ресурсу: https://google.com.ua

15. The Washington Post Fack Checker. [Електронний ресурс]. – Режим доступу до ресурсу: https://www.washingtonpost.com/news/fact-checker/

16. Snoopes. [Електронний ресурс]. – Режим доступу до ресурсу: https://www.snopes.com/fact-check/

17. Pilitifact [Електронний ресурс]. – Режим доступу до ресурсу: https://www.politifact.com/

18. Fiskkit [Електронний ресурс]. – Режим доступу до ресурсу: https://fiskkit.com/

19. Detecting Fake News with Sentiment Analysis and Network Metadata. / [Maniz Shrestha] // – 2019

20. Stance Prediction for Russian: Data and Analysis. / [Nikita Lozhnikov, Leon Derczynski, Manuel Mazzara] // – 2018

21. Was Charles Lieber Arrested for Selling the COVID-19 Coronavirus to China? [Електронний ресурс]. – Режим доступу до ресурсу: https://www.snopes.com/fact-check/charles-lieber-arrested-coronavirus/

22. Pilitifact [Електронний ресурс]. – Режим доступу до ресурсу: https://www.politifact.com/

23. The Onion [Електронний ресурс]. – Режим доступу до ресурсу: https://www.theonion.com/

24. Stance Prediction for Russian: Data and Analysis. / [Nikita Lozhnikov, Leon Derczynski, Manuel Mazzara] // – 2018

25. Brad Pitt Death Hoax [Електронний ресурс]. – Режим доступу до ресурсу: https://www.snopes.com/fact-check/brad-pitt-death-hoax/

26. Підхід до виявлення аномалій в потоках текстових даних. / [Ю.О. Олійник, О.Є. Афанасьєва, Г.Д. Аршакян] . // – 2020

27. Understanding bag-of-words model: A statistical framework . / [Yin Zhang] // – 2010

28. Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents. / [Shahzad Qaiser] // – 2018

29. Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents. / [Shahzad Qaiser] // – 2018

30. Using of Jaccard Coefficient for Keywords Similarity. / [Suphakit Niwattanakul, Jatsada Singthongchai, Ekkachai Naenudorn and Supachanun Wanapu] // – 2013

31. Правительство Украины увеличит финансирование патриотического кино за счет средств фонда борьбы с коронавирусом [Електронний ресурс]. – Режим доступу до ресурсу: https://ukraina.ru/news/20200408/1027329096.html

32. Semantic Cosine Similarity. / [Teruaki Kitasuka] // – 2012

33. Comparison Jaccard similarity, Cosine Similarity and Combined Both of the Data Clustering With Shared Nearest Neighbor Method. / [Lisna Zahrotun] // – 2016

34. В Україні за допомогу у висаджені городів штрафує на 17 тис грн [Електронний ресурс]. – Режим доступу до ресурсу: https://agronews.ua/news/shtrafy/

35. В Україні за допомогу у висаджені городів штрафує на 17 тис грн [Електронний ресурс]. – Режим доступу до ресурсу: https://www.unicorns.trade/v-ukraini-za-dopomohu-u-vysadzheni-horodiv-shtrafuie-na-17-tys-hrn/

36. В Україні за допомогу у висаджені городів штрафує на 17 тис грн [Електронний ресурс]. – Режим доступу до ресурсу: https://volyninfa.com.ua/news/2020/04/12/34899/

37. Pymorphy2 [Електронний ресурс]. – Режим доступу до ресурсу: https://pymorphy2.readthedocs.io/en/latest/

38. Dostoevsky [Електронний ресурс]. – Режим доступу до ресурсу: https://github.com/bureaucratic-labs/dostoevsky

39. Python [Електронний ресурс]. – Режим доступу до ресурсу: https://www.python.org/

40. Angular [Електронний ресурс]. – Режим доступу до ресурсу: https://angular.io/

41. Angular Material [Електронний ресурс]. – Режим доступу до ресурсу: https://material.angular.io/

42. OpenCorpora [Електронний ресурс]. – Режим доступу до ресурсу: http://opencorpora.org/

43. RuSentiment [Електронний ресурс]. – Режим доступу до ресурсу: https://github.com/text-machine-lab/rusentiment

44. textual-data-streaming-sentiment [Електронний ресурс]. – Режим доступу до ресурсу: https://github.com/YevheniiStepaniuk/textual-data-streaming-sentiment

ДОДАТОК А – КОД ПРОГРАМНОГО ЗАСТОСУНКУ

PythGenerator.py

```
from itertools import combinations
from pymorphy2 import MorphAnalyzer
from TextPreprocessor import sent_tokenizer_ru, word_tokenizer, stop_words_ru, sent_tokenizer_ua, stop_words_ua
import math
from TextGraph import rank_graph
from numpy import dot
from numpy.linalg import norm
from VectorMeasuresCalculator import tfidf, get_cosine, text_to_vector
import LSASummarizer

def similarity(s1, s2):
    n1 = norm(s1)
    n2 = norm(s2)
    if not n1 or not n2:
        return 0.0
    return dot(s1, s2)/(n1*n2)

def text_rank(text, language):
    sentences = []
    a = []
    if (language == 'ukrainian'):
```

```
        morph = MorphAnalyzer(lang='uk')
        sentences = sent_tokenizer_ua(text)
        if len(sentences) < 2:
            s = sentences[0]
            return [(1, 0, s)]
        a = tfidf(text, language, sent_tokenizer_ua, stop_words_ua)
    else:
        morph = MorphAnalyzer()
        sentences = sent_tokenizer_ru(text)
        if len(sentences) < 2:
            s = sentences[0]
            return [(1, 0, s)]
        a = tfidf(text, language, sent_tokenizer_ru, stop_words_ru)

    pairs = combinations(range(len(sentences)), 2)
    scores = [(i, j, similarity(a[i, :], a[j, :])) for i, j in pairs]
    scores = filter(lambda x: x[2], scores)

    pr = rank_graph(scores)

    return sorted(((i, pr[i], s) for i, s in enumerate(sentences) if i in pr),
                  key=lambda x: pr[x[0]], reverse=True)  # Сортировка по убыванию
ранга тройки
```

```
def summarize(text, language, n=5):
    tr = text_rank(text, language)
    top_n = sorted(tr[:n])
    text_rank_result = ' '.join(x[2] for x in top_n)
    lsa_result = LSASummarizer.summarize(text, language, n)
    cosine_text_rank = get_cosine(text_to_vector(text), text_to_vector(text_rank_result))
    cosine_lsa=get_cosine(text_to_vector(text), text_to_vector(lsa_result))
    if cosine_lsa > cosine_text_rank:
        return lsa_result
    return text_rank_result
```

LSASummarizer.py

```
from pymorphy2 import MorphAnalyzer
import numpy
from numpy.linalg import svd
from TextPreprocessor import sent_tokenizer_ru, word_tokenizer, stop_words_ru, sent_tokenizer_ua, stop_words_ua
import math


def create_dictionary(text, morph, stop_words):
    words = set(morph.parse(word)[0].normalized for word in word_tokenizer.tokenize(
        text.lower()) if word not in stop_words)
```

```
    return dict((w, i) for i, w in enumerate(words))



def create_matrix(text, sent_tokenizer, morph, dictionary):
    sentences = sent_tokenizer(text)

    words_count = len(dictionary)
    sentences_count = len(sentences)

    matrix = numpy.zeros((words_count, sentences_count))
    for col, sentence in enumerate(sentences):
        for word in word_tokenizer.tokenize(sentence.lower()):
            word = morph.parse(word)[0].normalized
            if word in dictionary:
                row = dictionary[word]
                matrix[row, col] += 1
    rows, cols = matrix.shape
    if rows and cols:
        word_count = numpy.sum(matrix)
        for row in range(rows):
            unique_word_count = numpy.sum(matrix[row, :])
            for col in range(cols):
                if matrix[row, col]:
                    matrix[row, col] = unique_word_count/word_count
```

```
    else:
        matrix = numpy.zeros((1, 1))
    return matrix


def compute_ranks(matrix, n):
    u_m, sigma, v_m = svd(matrix, full_matrices=False)
    powered_sigma = tuple(s**2 if i < n else 0.0 for i, s in enumerate(sigma))
    ranks = []
    for column_vector in v_m.T:
        rank = sum(s*v**2 for s, v in zip(powered_sigma, column_vector))
        ranks.append(math.sqrt(rank))
    return ranks


def summarize(text, language, n=5):
    morph = MorphAnalyzer()
    sent_tokenizer = sent_tokenizer_ru
    stop_words = stop_words_ru

    if language == 'ukrainian':
        morph = MorphAnalyzer(lang='uk')
        sent_tokenizer = sent_tokenizer_ua
        stop_words = stop_words_ua
```

```
    d = create_dictionary(text, morph, stop_words)
    m = create_matrix(text, sent_tokenizer, morph, d)
    r = compute_ranks(m, n)
    sentences = sent_tokenizer(text)
    rank_sort = sorted(((i, r[i], s) for i, s in enumerate(
        sentences)), key=lambda x: r[x[0]], reverse=True)
    top_n = sorted(rank_sort[:n])
    return ' '.join(x[2] for x in top_n)
```

RandomSummarizer.py

```
import random
from TextPreprocessor import sent_tokenizer_ru, sent_tokenizer_ua
def summarize(text, language, n):
    sent_tokenizer = sent_tokenizer_ru
    if language == 'ukrainian':
        sent_tokenizer = sent_tokenizer_ua
    sentences = sent_tokenizer(text)
    ratings = list(range(len(sentences)))
    random.shuffle(ratings)
    rand_sent = sorted((r, s) for r, s in zip(ratings, sentences))
    sent_n = rand_sent[:n]
    return ' '.join(s[1] for s in sent_n)
```

TextPreprocessor.py

```
# Инициализация всего необходимого для предобработки
from nltk.tokenize.punkt import PunktSentenceTokenizer, PunktParameters
from nltk.tokenize import RegexpTokenizer
import csv
import io

def get_stop_words(language):
    stopwords = []
    filename = 'stop_words_ru.csv'
    if language == 'ukrainian':
        filename = 'stop_words_ua.csv'
    with io.open(filename, 'r', encoding="utf-8") as file:
        for row in csv.reader(file):
            stopwords.append(row[0])
    return stopwords

def get_abbreviation(language):
    if language == 'ukrainian':
        return ['тис', 'грн', 'т.я', 'вул', 'сек', 'хв', 'обл', 'кв', 'пл', 'напр', 'гл', 'і.о', 'зам']
    return ['тыс', 'руб', 'т.е', 'ул', 'д', 'сек', 'мин', 'т.к', 'т.н', 'т.о', 'ср', 'обл', 'кв', 'пл',
            'напр', 'гл', 'и.о', 'им', 'зам', 'гл', 'т.ч']

punkt_param_ru = PunktParameters()
abbreviation_ru = get_abbreviation('russian')
```

```
punkt_param_ru.abbrev_types = set(abbreviation_ru)
sent_tokenizer_ru = PunktSentenceTokenizer(punkt_param_ru).tokenize
stop_words_ru = get_stop_words('russian')

punkt_param_ua = PunktParameters()
abbreviation_ua = get_abbreviation('ukrainian')
punkt_param_ua.abbrev_types = set(abbreviation_ua)
sent_tokenizer_ua = PunktSentenceTokenizer(punkt_param_ru).tokenize
stop_words_ua = get_stop_words('ukrainian')

word_tokenizer = RegexpTokenizer(r'\w+')
```

TextRankSummarizer.py

```
from itertools import combinations
from pymorphy2 import MorphAnalyzer
from TextPreprocessor import sent_tokenizer_ru, word_tokenizer, stop_words_ru, sent_tokenizer_ua, stop_words_ua
import math
from TextGraph import rank_graph

def similarity(s1, s2):
    if not len(s1) or not len(s2):
        return 0.0
    return len(s1.intersection(s2))/(math.log(len(s1)+1) + math.log(len(s2)+1))
```

```
def text_rank(text, language):
    sentences = []
    words = []

    if (language == 'ukrainian'):
        morph = MorphAnalyzer(lang='uk')
        sentences = sent_tokenizer_ua(text)
        if len(sentences) < 2:
            s = sentences[0]
            return [(1, 0, s)]
        words = [set(morph.parse(word)[0].normalized for word in word_tokenizer.tokenize(sentence.lower())
                     if word not in stop_words_ua) for sentence in sentences]
    else:
        morph = MorphAnalyzer()
        sentences = sent_tokenizer_ru(text)
        if len(sentences) < 2:
            s = sentences[0]
            return [(1, 0, s)]
        words = [set(morph.parse(word)[0].normalized for word in word_tokenizer.tokenize(sentence.lower())
                     if word not in stop_words_ru) for sentence in sentences]
```

```
    pairs = combinations(range(len(sentences)), 2)
    scores = [(i, j, similarity(words[i], words[j])) for i, j in pairs]
    scores = filter(lambda x: x[2], scores)
    pr = rank_graph(scores)

    return sorted(((i, pr[i], s) for i, s in enumerate(sentences) if i in pr),
                  key=lambda x: pr[x[0]], reverse=True)


def summarize(text, language, n=5):
    tr = text_rank(text, language)
    top_n = sorted(tr[:n])
    return ' '.join(x[2] for x in top_n)
```

```
import {Component, OnInit} from '@angular/core';
import { FormBuilder, FormGroup, FormControl, Validators } from '@angular/forms';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss']
})
export class AppComponent implements OnInit{
```

```
formGroup: FormGroup;
titleAlert: string = 'This field is required';
post: any = '';

constructor(private formBuilder: FormBuilder) { }

ngOnInit() {
  this.createForm();
}

createForm() {
  this.formGroup = this.formBuilder.group({
    language: '',
    date: '',
    text: ''
  });
}

get name() {
  return this.formGroup.get('name') as FormControl
}

onSubmit(post) {
```

```
  this.post = post;
 }

}
<mat-toolbar color="primary">
 <span class="fill-remaining-space">Пошук шляху розповсюдження дезінформації</span>
</mat-toolbar>

<div class="container" *ngIf="!post" novalidate>
 <form [formGroup]="formGroup" (ngSubmit)="onSubmit(formGroup.value)" class="form">

   <mat-form-field class="form-element">
    <mat-select placeholder="Мова публікації" formControlName="language">
     <mat-option *ngFor="let lang of ['Українська', 'Російська']" [value]="lang">
      {{lang}}
     </mat-option>
    </mat-select>

   </mat-form-field>

   <mat-form-field class="form-element">
    <input matInput placeholder="Дата публікації" formControlName="date">
```

```
    </mat-form-field>

    <mat-form-field class="form-element">
     <textarea matInput placeholder="Текст публікації" matTextareaAutosize matAutosizeMinRows="20" matAutosizeMaxRows="20" formControlName="text"></textarea>

    </mat-form-field>

    <div class="form-element">
     <button mat-raised-button color="primary" type="submit" class="button" [disabled]="!formGroup.valid">Перевірити</button>
    </div>

  </form>
</div>
.fill-remaining-space {
  /* This fills the remaining space, by using flexbox.
     Every toolbar row uses a flexbox row layout. */
  flex: 1 1 auto;
}
```

```
.container {
  padding: 10px;
  background-color: #303030;
  height: 100%;
}

.form {
  min-width: 150px;
  max-width: 500px;
  width: 100%;
  margin: 0 auto;
}

.form-element {
  padding: 5px 0px 25px 2px;
  width: 100%;
}

.button {
  width: 100%;
}

:host {
```

```
  background-color: rgb(48, 48, 48);
  color: white;
}
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { NoopAnimationsModule } from '@angular/platform-browser/animations';
import {MatInputModule} from '@angular/material/input';
import {ReactiveFormsModule} from '@angular/forms';
import {MatFormFieldModule} from '@angular/material/form-field';
import {MaterialModule} from './material.module';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    NoopAnimationsModule,
    ReactiveFormsModule,
    MaterialModule,
```

```
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';


const routes: Routes = [];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
import { NgModule } from '@angular/core';

import { MatAutocompleteModule } from '@angular/material/autocomplete';
import { MatButtonModule } from '@angular/material/button';
import { MatCheckboxModule } from '@angular/material/checkbox';
import { MatChipsModule } from '@angular/material/chips';
import { MatNativeDateModule } from '@angular/material/core';
import { MatDatepickerModule } from '@angular/material/datepicker';
```

```
import { MatDialogModule } from '@angular/material/dialog';
import { MatExpansionModule } from '@angular/material/expansion';
import { MatIconModule } from '@angular/material/icon';
import { MatInputModule } from '@angular/material/input';
import { MatListModule } from '@angular/material/list';
import { MatMenuModule } from '@angular/material/menu';
import { MatProgressBarModule } from '@angular/material/progress-bar';
import { MatProgressSpinnerModule } from '@angular/material/progress-spinner';
import { MatRadioModule } from '@angular/material/radio';
import { MatSelectModule } from '@angular/material/select';
import { MatSidenavModule } from '@angular/material/sidenav';
import { MatSlideToggleModule } from '@angular/material/slide-toggle';
import { MatSliderModule } from '@angular/material/slider';
import { MatSnackBarModule } from '@angular/material/snack-bar';
import { MatStepperModule } from '@angular/material/stepper';
import { MatTabsModule } from '@angular/material/tabs';
import { MatTooltipModule } from '@angular/material/tooltip';
import {DragDropModule} from "@angular/cdk/drag-drop";
import {MatToolbarModule} from '@angular/material/toolbar';

@NgModule({
  exports: [
    MatIconModule,
    MatButtonModule,
```

```
    MatDatepickerModule,
    MatProgressBarModule,
    MatRadioModule,
    MatInputModule,
    MatDialogModule,
    MatTooltipModule,
    MatCheckboxModule,
    MatSliderModule,
    MatNativeDateModule,
    MatMenuModule,
    MatProgressSpinnerModule,
    MatSelectModule,
    MatTabsModule,
    MatSnackBarModule,
    MatStepperModule,
    MatAutocompleteModule,
    MatChipsModule,
    MatExpansionModule,
    MatListModule,
    MatSlideToggleModule,
    MatSidenavModule,
    DragDropModule,
    MatToolbarModule,
  ],
```

```
})
export class MaterialModule {


}
```

ПЛАКАТ 1 ДІАГРАМА ПОСЛІДОВНОСТЕЙ КЛАСИФІКАЦІЇ ТЕКСТУ НА НАЯВНІСТЬ ЕЛЕМЕНТІВ ДЕЗІНФОРМАЦІЇ

ПЛАКАТ 2 ДІАГРАМА РОЗГОРТАННЯ

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Автоматизованих систем обробки інформації і управління

Рівень вищої освіти – другий (магістерський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-наукова програма - «Інженерія програмного забезпечення комп'ютеризованих систем»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Олександр ПАВЛОВ
«___»_____20__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Ошийку Ярославу Романовичу

1. Тема дисертації «*Математичне та програмне забезпечення виявлення елементів дезінформації в потоках текстових даних*», науковий керівник дисертації Олійник Юрій Олександрович, старший викладач, затверджені наказом по університету від «24» березня 2020 р. №910-с

2. Термін подання студентом дисертації *27.04.2020*

3. Об'єкт дослідження *процеси виявлення елементів дезінформації в текстових потоках даних*

4. Предмет дослідження *методи виявлення елементів дезінформації в текстових потоках даних*

5. Перелік завдань, які потрібно розробити *виконати аналіз існуючих алгоритмів та методів комп'ютерної лінгвістики та машинного навчання для класифікації текстових потоків даних та виявлення елементів дезінформації, розробити алгоритм первинної обробки тексту для збільшення точності визначення елементів дезінформації, розробити метод виявлення елементів дезінформації в текстових потоках даних, виконати програмну реалізацію розробленого методу виявлення елементів дезінформації в текстових потоках даних, провести аналіз отриманих результатів для оцінки якості, провести дослідження ефективності алгоритму*

6. Орієнтовний перелік графічного (ілюстративного) матеріалу Діаграма послідовності класифікації дезінформації

7. Орієнтовний перелік публікацій *Задача виявлення дезінформації в текстових потоках даних, Метод виявлення елементів дезінформації в текстових потоках даних*

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	*Огляд існуючих підходів до виявлення елементів дезінформації*	*30.11.2018*	
2	*Постановка та формалізація математичної моделі задачі*	*15.03. 2019*	
3	*Розробка методу виявлення елементів дезінформації*	*10.10.2019*	
4	*Проведення експериментальних досліджень розробленого алгоритму*	*20.12.2019*	
5	*Оформлення документації*	*19.03.2020*	
6	*Подання дисертації на попередній захист*	*23.04.2020*	
7	*Подання дисертації на основний захист*	*19.05.2020*	

Студент Ошийко Ярослав

Науковий керівник Юрій Олійник